Algorithms for Sparse Signal Recovery in Compressed Sensing

Aqib Ejaz

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, 19 May 2015

Thesis supervisor:

Prof. Visa Koivunen

Thesis advisor:

D.Sc. (Tech.) Esa Ollila

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Aqib Ejaz

Title: Algorithms for Sparse Signal Recovery in Compressed Sensing

Date: 19 May 2015 Language: English Number of pages: 9+65

Department of Signal Processing and Acoustics

Professorship: Signal Processing

Supervisor: Prof. Visa Koivunen

Advisor: D.Sc. (Tech.) Esa Ollila

Compressed sensing and sparse signal modeling have attracted considerable research interest in recent years. The basic idea of compressed sensing is that by exploiting the sparsity of a signal one can accurately represent the signal using fewer samples than those required with traditional sampling. This thesis reviews the fundamental theoretical results in compressed sensing regarding the required number of measurements and the structure of the measurement system. The main focus of this thesis is on algorithms that accurately recover the original sparse signal from its compressed set of measurements. A number of greedy algorithms for sparse signal recovery are reviewed and numerically evaluated. Convergence properties and error bounds of some of these algorithms are also reviewed. The greedy approach to sparse signal recovery is further extended to multichannel sparse signal model. A widely-used non-Bayesian greedy algorithm for the joint recovery of multichannel sparse signals is reviewed. In cases where accurate prior information about the unknown sparse signals is available, Bayesian estimators are expected to outperform non-Bayesian estimators. A Bayesian minimum mean-squared error (MMSE) estimator of the multichannel sparse signals with Gaussian prior is derived in closed-form. Since computing the exact MMSE estimator is infeasible due to its combinatorial complexity, a novel algorithm for approximating the multichannel MMSE estimator is developed in this thesis. In comparison to the widely-used non-Bayesian algorithm, the developed Bayesian algorithm shows better performance in terms of mean-squared error and probability of exact support recovery. The algorithm is applied to direction-of-arrival estimation with sensor arrays and image denoising, and is shown to provide accurate results in these applications.

Keywords: compressed sensing, sparse modeling, greedy algorithms, MMSE, Bayesian, multichannel sparse recovery.

Preface

The work I have presented in this thesis was carried out at the Department of Signal Processing & Acoustics, Aalto University during a period of eight months starting in June of 2014. The aim of this work was to develop a solid understanding of compressed sensing that would result in an original contribution to this rapidly growing field. This involved a lot of literature review and computer simulations. The final outcome of all that hard work is RandSOMP, a novel Bayesian algorithm developed in this thesis for recovering multichannel sparse signals in compressed sensing.

This work would not be possible without the continuous help and guidance that I received from my supervisor, Prof. Visa Koivunen, and my advisor, D.Sc. (Tech) Esa Ollila. I thank them for trusting in my abilities and giving me the opportunity to work in their research group. I owe special thanks to Esa Ollila for his deep involvement with this work. His insightful research ideas and helpful comments have mainly guided the course of this work.

Finally, I would like to thank my family and friends for their love and support. Your presence makes my life richer.

Otaniemi, 19 May 2015

Aqib Ejaz

Contents

Symbols

m	A scalar
$\mathbf{x}$	A vector
$\mathbf{A}$	A matrix
$\mathbf{x}^{\mathrm{T}}$	Transpose of $\mathbf{x}$
$\mathbf{x}^{\mathrm{H}}$	Conjugate transpose of $\mathbf{x}$
$\mathbb{R}^n$	The n-dimensional real vector space
$\mathbb{C}^n$	The n-dimensional complex vector space
$\|\cdot\|_0$	ℓ_0-pseudonorm
$\|\cdot\|_1$	ℓ_1-norm
$\|\cdot\|_2$	Euclidean norm
$\|\cdot\|_F$	Frobenius norm
$\otimes$	Kronecker product
$\mathrm{vec}(\cdot)$	Vectorization operation
$\mathbb{E}[\cdot]$	Expected value
$\mathrm{Tr}(\cdot)$	Trace of a square matrix

Abbreviations

CoSaMP	Compressive sampling matching pursuit
DOA	Direction-of-arrival
FBMP	Fast Bayesian matching pursuit
IHT	Iterative hard thresholding
MMSE	Minimum mean-squared error
MMV	Multiple measurement vectors
MSE	Mean-squared error
MUSIC	Multiple signal classification
NSP	Null space property
OMP	Orthogonal matching pursuit
PER	Probability of exact support recovery
PSNR	Peak signal-to-noise ratio
RandCoSaMP	Randomized compressive sampling matching pursuit
RandIHT	Randomized iterative hard thresholding
RandOMP	Randomized orthogonal matching pursuit
RandSOMP	Randomized simultaneous orthogonal matching pursuit
RIP	Restricted isometry property
RMSE	Root mean-squared error
SMV	Single measurement vector
SNR	Signal-to-noise ratio
SOMP	Simultaneous orthogonal matching pursuit
ULA	Uniform linear array
WRS	Weighted random sampling

List of Figures

Chapter 1

Introduction

1.1 Background

In signal processing, the Nyquist-Shannon sampling theorem has long been seen as the guiding principle for signal acquisition. The theorem states that an analog signal can be perfectly reconstructed from its samples if the samples are taken at a rate at least twice the bandwidth of the signal. This gives a sampling rate which is sufficient to reconstruct an analog signal from its samples. But is this sampling rate necessary as well? The recently developed field of compressed sensing [1, 2, 3, 4, 5] has successfully answered that question. It turns out that it is possible to sample a signal at a lower than Nyquist rate without any significant loss of information. In order to understand when and how this is possible, let us start with an example.

Consider a digital camera that acquires images with a 20 megapixel resolution. A raw image produced by this camera would require about 60 megabytes (MB) of storage space. Typically, images are compressed using some compression technique (e.g., JPEG standard) which often results in a large reduction in the size of the image. The same 60 MB raw image, for example, could be compressed and stored in a 600 kilobytes storage space without any significant loss of information. The key fact that enables this compression is the redundancy of information in the image. If one could exploit this information redundancy then it should be possible to sample at a lower rate. The real challenge here is to know what is important in a signal before the signal is even sampled. The theory of compressed sensing provides a surprising solution to that challenge, i.e., the solution lies in the randomly weighted sums of signal measurements.

In compressed sensing, the signal to be sampled is generally represented as a vector, say $\mathbf{x} \in \mathbb{R}^n$. The information contained in $\mathbf{x}$ is assumed to be highly redundant. This means $\mathbf{x}$ can be represented in an orthonormal basis system $\mathbf{W} \in \mathbb{R}^{n \times n}$ using $k \ll n$ basis vectors, i.e., $\mathbf{x} = \mathbf{W}\mathbf{s}$. The vector $\mathbf{s} \in \mathbb{R}^n$ has only k non-zero elements and is called the sparse representation of $\mathbf{x}$. A linear sampling process is represented by a measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. The sampled signal is represented as a vector $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v}$, where $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^m$ represents additive noise in the system. The goal in compressed sensing is to acquire the signal $\mathbf{x}$ with no or insignificant loss of information using fewer than n measurements, i.e., to accurately recover $\mathbf{x}$ from

$\mathbf{y}$ with $m < n$. In other words, the central theme of compressed sensing is about finding a sparse solution to an underdetermined linear system of equations.

A fundamental question in compressed sensing is about how small the number of measurements m can be made in relation to the dimension n of the signal. The role played in the sampling process by the sparsity k of the signal is also worth exploring. Theoretical results in compressed sensing prove that the minimum number of measurements m needed to capture all the important information in a sparse signal grows linearly with k and only logarithmically with n [2]. This makes sense since the actual amount of information in the original signal is indicated by k and not by n, which is why m has a stronger linear dependence on k but a weaker logarithmic dependence on n.

The design of the measurement matrix $\mathbf{A}$ is also an important aspect of compressed sensing. It is desired that $\mathbf{A}$ is not dependent on the signal being sampled so that it can be applied to any sparse signal regardless of the actual contents of the signal. As it turns out, $\mathbf{A}$ can be chosen independent of the sampled signal as long as it is different from the basis $\mathbf{W}$ in which the signal is known to be sparse. Theoretical results in compressed sensing have shown that random matrices that possess the restricted isometry property and/or have low mutual coherence among their columns are suitable candidates for this purpose [1, 2].

The recovery of the signal $\mathbf{x}$ (or equivalently $\mathbf{s}$) from its compressed measurements $\mathbf{y}$ has gained a lot of attention in the past several years. A number of techniques have been developed for this purpose. These can be broadly classified into two categories. First of these is the set of techniques that are based on ℓ_1-norm minimization using convex optimization [6, 7, 8]. These techniques generally have excellent recovery performance and also have guaranteed performance bounds. High computational cost is one major drawback of these techniques, which makes it difficult to apply these techniques to large-scale problems. Alternative techniques include iterative greedy methods [9, 10, 11, 12]. These methods make sub-optimal greedy choices in each iteration and hence are computationally much more efficient. They may suffer a little in their recovery performance but are much more useful in large-scale problems. These methods generally have proven performance bounds, making them a reliable set of tools for sparse signal recovery.

1.2 Research Problem

The main focus of this thesis is the study of signal recovery algorithms in compressed sensing. Specifically, greedy algorithms that estimate the sparse signals in both Bayesian and non-Bayesian frameworks are considered in this thesis. The objective is to give a performance evaluation of several greedy algorithms under different parameter settings. Multichannel sparse signal recovery problem is also considered and a novel Bayesian algorithm is developed for solving this problem.

1.3 Contributions of the Thesis

The following are the main contributions of this thesis:

- Several existing theoretical results regarding the measurement system and the recovery algorithms in compressed sensing are reviewed in detail in Chapters 2-3. This includes the results on the minimum number of measurements required for accurate signal recovery, structure of the measurement matrix, and error bounds of some of the recovery algorithms.

- Several greedy algorithms for sparse signal recovery are reviewed in Chapters 3-4 and their codes are developed in Matlab. Based on simulations carried out in Matlab, the signal recovery performance of these algorithms is numerically evaluated and compared.

- A novel Bayesian algorithm for multichannel sparse signal recovery is developed as the main contribution of this thesis in Chapter 5. A generalization of the developed algorithm for complex-valued signals is developed in Chapter 6. The performance of the algorithm is numerically evaluated in Matlab. The algorithm is applied to direction-of-arrival (DOA) estimation with sensor arrays and image denoising, and is shown to produce accurate results in these applications. The algorithm and its derivation are also described in a conference paper that has been submitted for publication [13].

1.4 Outline of the Thesis

The thesis is organized into eight chapters. An introduction to this thesis is given in Chapter 1. In Chapter 2, we present several basic definitions. Some fundamental theoretical results in compressed sensing are also presented. Chapter 3 reviews some of the commonly used greedy algorithms for signal recovery in compressed sensing. In Chapter 4, we review several Bayesian algorithms that take prior knowledge about the sparse signal into account. We numerically evaluate signal recovery performance of several Bayesian and non-Bayesian methods in this chapter. In Chapter 5, we present the problem of multichannel sparse signal recovery and develop a novel Bayesian algorithm for solving it. The performance of the developed algorithm is numerically evaluated and compared with a widely-used greedy algorithm. Chapter 6 generalizes the developed algorithm for recovering multichannel complex-valued sparse signals. In Chapter 7, we apply the algorithms for multichannel sparse signal recovery in DOA estimation with sensor arrays and image denoising. Concluding remarks of the thesis are given in Chapter 8.

Chapter 2

Compressed Sensing

In this chapter we will formulate the problem of sparse signal recovery in compressed sensing. We will give several basic definitions. We will also review important theoretical results in compressed sensing concerning the minimum number of measurements required for accurate signal recovery and the structure of the measurement matrix. We start the chapter with a formal mathematical formulation of the compressed sensing model.

Let $\mathbf{x} \in \mathbb{R}^n$ be an unknown signal vector. The vector $\mathbf{x}$ can be assumed to be sparse (i.e., most of the entries are zero) either in the canonical basis or some other non-canonical orthonormal basis. For simplicity of representation, $\mathbf{x}$ is assumed to be sparse in the canonical basis. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a known measurement matrix with $m < n$. Observed vector $\mathbf{y} \in \mathbb{R}^m$ is then modeled as

$$\mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2.1}$$

It is then required to recover $\mathbf{x}$ given the knowledge of just $\mathbf{A}$ and $\mathbf{y}$. Since it is assumed that $\mathbf{A}$ has fewer rows than columns (underdetermined system), dimension of the null space of $\mathbf{A}$ will be greater than zero ($NullSpace(\mathbf{A}) = \{\mathbf{z} \in \mathbb{R}^n : \mathbf{A}\mathbf{z} = \mathbf{0}\}$). As a consequence, the mapping from $\mathbf{x}$ to $\mathbf{y}$ is non-injective since for every $\mathbf{x}$ there are infinitely many vectors of the form $\mathbf{x} + \mathbf{z}$ ($\mathbf{z} \in NullSpace(\mathbf{A}) \backslash \{\mathbf{0}\}$) such that $\mathbf{A}(\mathbf{x}+\mathbf{z}) = \mathbf{y}$. Therefore, in general, it is not possible to recover $\mathbf{x}$ from $\mathbf{y}$ unless there are additional conditions which are satisfied by $\mathbf{x}$. Sparsity is one such condition. In practice, most of the natural signals are either sparse or well approximated by sparse signals. Formally, a vector is said to be *k-sparse* if at most k of its components are non-zero. When $\mathbf{x}$ is known to be sparse then it becomes possible to recover it in an underdetermined framework. Besides sparsity of $\mathbf{x}$, the structure of the measurement matrix $\mathbf{A}$ is also crucial for accurate recovery. This will be discussed in greater detail later on, but first we introduce basic terminology.

Definition 1. *Cardinality of a finite set S is defined as 'the number of elements of the set'. Cardinality of S is indicated by $|S|$.*

Definition 2. *For a real number $p \geq 1$, ℓ_p-norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as*

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p},$$

where $|x_i|$ indicates the absolute value of the i-th component of $\mathbf{x}$.

Two commonly used ℓ_p-norms are the ℓ_1-norm and the ℓ_2-norm:

$$\|\mathbf{x}\|_1 := \sum_{i=1}^{n} |x_i|,$$
$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} |x_i|^2}.$$

For $0 < p < 1$, ℓ_p-norm as defined above does not satisfy the triangle inequality and therefore cannot be called a norm. Instead it is called a quasinorm as it satisfies the following weaker version of the triangle inequality,

$$\|\mathbf{x} + \mathbf{y}\|_p \leq c \left(\|\mathbf{x}\|_p + \|\mathbf{y}\|_p\right),$$

with $c = 2^{1/p-1}$.

Definition 3. *The ℓ_0-pseudonorm of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as*

$$\|\mathbf{x}\|_0 := \Big|\{j : x_j \neq 0\}\Big|,$$

i.e., the ℓ_0-pseudonorm of a vector is defined as the number of non-zero components of the vector.

The ℓ_0-pseudonorm is not a proper norm as it does not satisfy the homogeneity property, i.e.,

$$\|c\mathbf{x}\|_0 \neq |c|\|\mathbf{x}\|_0,$$

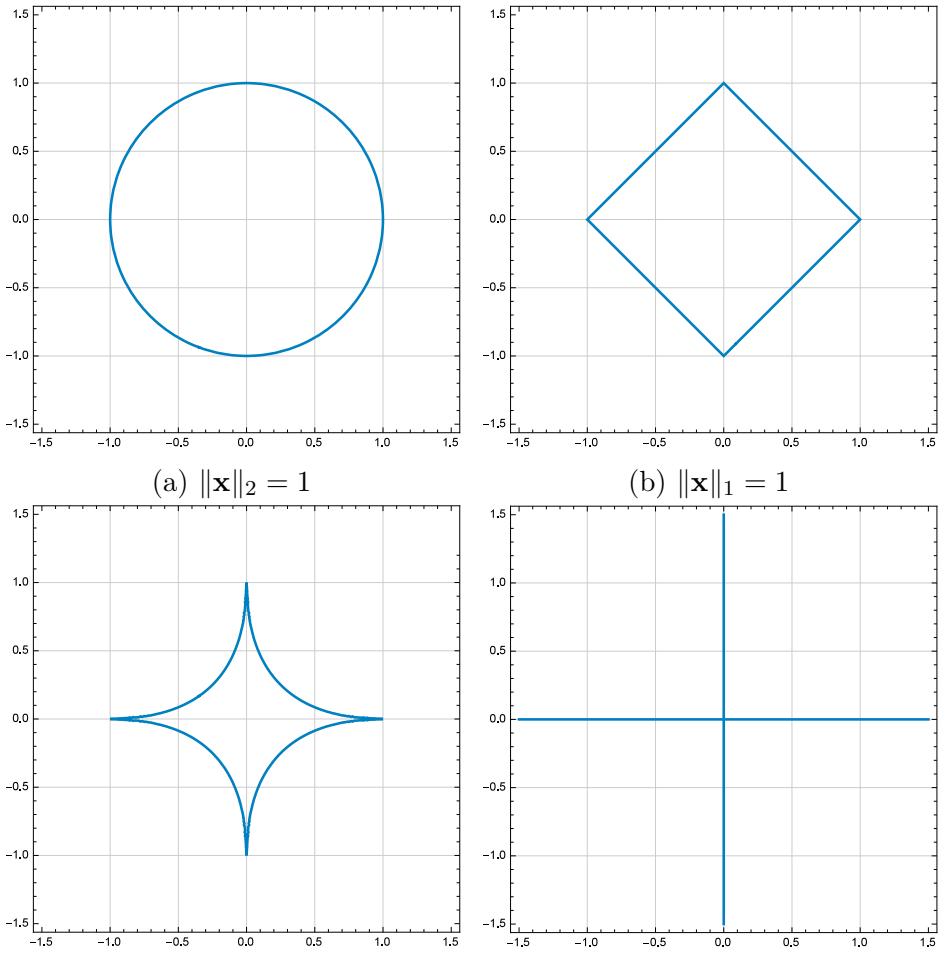
for all scalars $c \neq \pm 1$. The ℓ_0-pseudonorm and ℓ_p-norm with three different values of p are illustrated in Figure 2.1.

Definition 4. *Spark of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as 'the cardinality of the smallest set of linearly dependent columns of $\mathbf{A}$', i.e.,*

$$Spark(\mathbf{A}) = \min |V| \quad \text{such that} \quad V \subseteq [n] \text{ and } rank(\mathbf{A}_V) < |V|,$$

where $[n] = \{1, 2, \ldots, n\}$ and $\mathbf{A}_V$ is the matrix consisting of those columns of $\mathbf{A}$ which are indexed by V.

For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, every set of $m + 1$ columns of $\mathbf{A}$ is guaranteed to have linearly dependent columns. On the other hand, one needs to take at least two column vectors to form a linearly dependent set. Therefore, $2 \leq Spark(\mathbf{A}) \leq m + 1$. The definition of the spark of a matrix draws some parallels to the definition of the rank of a matrix, i.e., spark is the cardinality of the smallest set of linearly dependent columns whereas rank is the cardinality of the largest set of linearly independent columns. While rank of a matrix can be computed efficiently, computing spark of a matrix is NP-Hard [14].



(a) $\|\mathbf{x}\|_2 = 1$ (b) $\|\mathbf{x}\|_1 = 1$

(c) $\|\mathbf{x}\|_{1/2} = 1$ (d) $\|\mathbf{x}\|_0 = 1$

Figure 2.1: Unit Balls in $\mathbb{R}^2$

From the definition of the spark of a matrix $\mathbf{A}$, it can be concluded that $\|\mathbf{z}\|_0 \geq Spark(\mathbf{A})$ for all $\mathbf{z} \in NullSpace(\mathbf{A})\backslash\{\mathbf{0}\}$. Now suppose that $\mathbf{x}$ is k-sparse and $\mathbf{A}$ is designed so that $Spark(\mathbf{A}) \geq 2k+1$. Then by solving the optimization problem,

$$\min \ \|\mathbf{x}\|_0 \qquad \text{subject to} \ \ \mathbf{A}\mathbf{x} = \mathbf{y}, \tag{P_0}$$

one is guaranteed to recover $\mathbf{x}$ exactly. It is quite easy to show that above formulation indeed recovers $\mathbf{x}$ unambiguously. Since $\|\mathbf{x}\|_0 \leq k$ and $\|\mathbf{z}\|_0 \geq Spark(\mathbf{A}) \geq 2k+1$ for all $\mathbf{z} \in NullSpace(\mathbf{A})\backslash\{\mathbf{0}\}$ it follows that $\|\mathbf{x}+\mathbf{z}\|_0 \geq k+1$ and therefore $\|\mathbf{x}\|_0 < \|\mathbf{x}+\mathbf{z}\|_0$. In other words, $\mathbf{x}$ is sparser than all other solutions $(\mathbf{x}+\mathbf{z})$ of (2.1) and thus solving (P_0) will recover $\mathbf{x}$ and not any other solution.

So far it has been assumed that there was no noise present in the system which is not a realistic assumption when real-world data or physical observations are processed. A more realistic model in which the observations are contaminated by an additive noise is given as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v}, \tag{2.2}$$

where $\mathbf{v} \in \mathbb{R}^m$ is the unknown noise vector. To compensate for the effect of noise in (2.2) the optimization problem (P_0) is slightly modified as

$$\min \ \|\mathbf{x}\|_0 \qquad \text{subject to} \ \ \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \eta, \tag{$\mathrm{P}_{0,\eta}$}$$

where η is a measure of the noise power. In ($\mathrm{P}_{0,\eta}$) the objective is to find the sparsest vector $\mathbf{x}$ such that ℓ_2-norm of the error $\mathbf{y} - \mathbf{A}\mathbf{x}$ remains within a certain bound η.

Minimizing ℓ_0-pseudonorm leads to exact recovery in noiseless case (provided $\mathbf{A}$ is appropriately chosen) but this approach suffers from one major drawback. It turns out that both (P_0) and ($\mathrm{P}_{0,\eta}$) are NP-hard problems [15] so in general there is no known algorithm which can give a solution to these problems in polynomial time and exhaustive search is the only option available which is not practical even in fairly simple cases.

To circumvent this limitation several approaches have been proposed in the literature. One such popular approach is to solve the optimization problems in which ℓ_1-norm replaces the ℓ_0-pseudonorm, i.e.

$$\min \ \|\mathbf{x}\|_1 \qquad \text{subject to} \ \ \mathbf{A}\mathbf{x} = \mathbf{y}, \tag{P_1}$$

$$\min \ \|\mathbf{x}\|_1 \qquad \text{subject to} \ \ \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \eta. \tag{$\mathrm{P}_{1,\eta}$}$$

(P_1) is referred to as *basis pursuit* while ($\mathrm{P}_{1,\eta}$) is known as *quadratically-constrained basis pursuit* [7]. Since these are convex optimization problems, there are methods that can efficiently solve them. The only question that remains is whether the solution obtained through ℓ_1-minimization is relevant to the original ℓ_0-minimization problem and the answer to that is in the affirmative. In fact, if $\mathbf{A}$ satisfies the so-called *null space property (NSP)* [16] then solving (P_1) will recover the same solution as the one obtained by solving (P_0).

Definition 5. *An $m \times n$ matrix $\mathbf{A}$ is said to satisfy the null space property of order k if the following holds for all $\mathbf{z} \in NullSpace(\mathbf{A}) \backslash \{\mathbf{0}\}$ and for all $S \subset [n] = \{1, 2, \ldots, n\}$ with $|S| = k$,*

$$\|\mathbf{z}_S\|_1 < \|\mathbf{z}_{\overline{S}}\|_1,$$

where $\mathbf{z}_S$ is the vector consisting of only those components of $\mathbf{z}$ which are indexed by S while $\mathbf{z}_{\overline{S}}$ is the vector consisting of only those components of $\mathbf{z}$ which are indexed by $\overline{S} = [n] \backslash S$.

If a matrix $\mathbf{A}$ satisfies NSP of order k then it is quite easy to see that for every k-sparse vector $\mathbf{x}$ and for every $\mathbf{z} \in NullSpace(\mathbf{A}) \backslash \{\mathbf{0}\}$ it holds that $\|\mathbf{x}\|_1 < \|\mathbf{x} + \mathbf{z}\|_1$. This means that the k-sparse solution $\mathbf{x}$ to (2.1) will have smaller ℓ_1-norm than all other solutions of the form $(\mathbf{x} + \mathbf{z})$ and therefore by solving (P_1) one will recover $\mathbf{x}$ unambiguously. Furthermore, if a matrix satisfies NSP of order k then it implies that the spark of the matrix will be greater than $2k$. Therefore it is very important that the measurement matrix satisfies NSP. It turns out that there exists an effective methodology that generates measurement matrices which satisfy NSP. Before proceeding, it is important to first introduce the concept of *restricted isometry property (RIP)* [3, 17].

Definition 6. *An $m \times n$ matrix $\mathbf{A}$ is said to satisfy the restricted isometry property of order $2k$ if the following holds for all $2k$-sparse vectors $\mathbf{x}$ with $0 < \delta_{2k} < 1$,*

$$(1 - \delta_{2k})\|\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq (1 + \delta_{2k})\|\mathbf{x}\|_2^2,$$

where δ_{2k} is called the restricted isometry constant.

A matrix that satisfies RIP of order $2k$ will operate on k-sparse vectors in a special way. Consider two k-sparse vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ and a measurement matrix $\mathbf{A}$ that satisfies RIP of order $2k$. Now $\mathbf{x}_1 - \mathbf{x}_2$ will be $2k$-sparse and since $\mathbf{A}$ is assumed to satisfy RIP of order $2k$ the following will hold,

$$(1 - \delta_{2k})\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \leq \|\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2)\|_2^2 \leq (1 + \delta_{2k})\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2,$$

$$\Rightarrow (1 - \delta_{2k})\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 \leq \|\mathbf{A}\mathbf{x}_1 - \mathbf{A}\mathbf{x}_2\|_2^2 \leq (1 + \delta_{2k})\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2.$$

This means that the distance between $\mathbf{A}\mathbf{x}_1$ and $\mathbf{A}\mathbf{x}_2$ is more or less the same as the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$, which implies that no two distinct k-sparse vectors are mapped by $\mathbf{A}$ to the same point. As a consequence, it can be argued that RIP of order $2k$ implies that the spark of the matrix is greater than $2k$. Furthermore, it can be proved that RIP of order $2k$ with $0 < \delta_{2k} < 1/(1 + \sqrt{2})$ implies NSP of order k [18]. Therefore for accurate recovery of sparse signals it is sufficient that the measurement matrix satisfies RIP. Theoretical results in compressed sensing show that an $m \times n$ matrix will satisfy RIP for k-sparse vectors with a reasonably high probability if,

$$m = \mathcal{O}\left(k \log \frac{n}{k}\right),$$

and entries of the matrix are taken independent of each other from a Gaussian or a Bernoulli distribution [19]. This gives a simple but effective method of constructing measurement matrices that enable accurate recovery of sparse signals.

In addition to ℓ_1-minimization, there exist greedy methods for sparse signal recovery in compressed sensing. Greedy pursuit methods and thresholding-based methods are two broad classes of these algorithms which are generally faster to compute than ℓ_1-minimization methods. *Orthogonal matching pursuit (OMP)* [9, 10] and *compressive sampling matching pursuit (CoSaMP)* [12] are two of the well-known greedy pursuit methods whereas *iterative hard thresholding (IHT)* and its many variants [11, 20, 21] are examples of thresholding-based methods. Some of these algorithms are discussed in detail in the next chapter.

Chapter 3

Non-Bayesian Greedy Methods

Sparse signal recovery based on ℓ_1-minimization is an effective methodology which, under certain conditions, can result in an exact signal recovery. In addition, ℓ_1-minimization also has very good performance guarantees which make it a reliable tool for sparse signal recovery. One drawback of the methods based on ℓ_1-minimization is their higher computational cost in large-scale problems. Therefore, algorithms that scale up better and are similar in performance in comparison to the convex optimization methods are needed. Greedy algorithms described in this chapter are good examples of such methods [9, 10, 11, 12, 20, 21]. These algorithms make significant savings in computation by performing locally optimal greedy iterations. Some of these methods also have certain performance guarantees that are somewhat similar to the guarantees for ℓ_1-minimization. All of this makes greedy algorithms an important set of tools for recovering sparse signals.

The greedy algorithms reviewed in this chapter recover the unknown sparse vector in a non-Bayesian framework, i.e., the sparse vector is treated as a fixed unknown and no prior assumption is made about the probability distribution of the sparse vector. In contrast, the Bayesian methods [22, 23, 24] reviewed in Chapter 4 consider the unknown sparse vector to be random and assume that some prior knowledge of the signal distribution is available. Non-Bayesian greedy methods are computationally very simple. There are also some theoretical results associated with non-Bayesian greedy methods that provide useful guarantees regarding the convergence and error bounds of these methods.

Bayesian methods do not have any theoretical guarantees regarding their recovery performance. They also require some kind of greedy approximation scheme to circumvent their combinatorial complexity. Their use for sparse signal recovery is justified only when some accurate prior knowledge of the signal distribution is available. In such cases, Bayesian methods can incorporate the prior knowledge of the signal distribution into the estimation process and hence provide better recovery performance than non-Bayesian methods.

3.1 Orthogonal Matching Pursuit

Consider a slightly modified version of the optimization problem ($\mathrm{P}_{0,\eta}$) in which role of the objective function is interchanged with the constraint function, i.e.,

$$\min \ \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 \qquad \text{subject to} \ \ \|\mathbf{x}\|_0 \leq k. \tag{P2}$$

In (P2) the objective is to minimize the squared ℓ_2-norm of the residual error $\mathbf{y}-\mathbf{A}\mathbf{x}$ subject to the constraint that $\mathbf{x}$ should be k-sparse. Let us define *support set* of a vector $\mathbf{x}$ as *the set of those indices where* $\mathbf{x}$ *has non-zero components*, i.e.,

$$\mathrm{supp}(\mathbf{x}) = \{j : x_j \neq 0\},$$

where x_j denotes j-th component of $\mathbf{x}$. Assuming $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{x} \in \mathbb{R}^n$, the total number of different supports that k-sparse $\mathbf{x}$ can have is given by $L = \sum_{i=1}^{k} \binom{n}{i}$. In theory one could solve (P2) by going through all L supports one by one and for each support finding the optimal solution by projecting $\mathbf{y}$ onto the space spanned by those columns of $\mathbf{A}$ which are indexed by the selected support. This is equivalent to solving a least squares problem for each possible support of $\mathbf{x}$. Finally one could compare all L solutions and select the one that gives the smallest squared ℓ_2-norm of the residual. Although least squares problem can be solved efficiently, difficulty with the given brute-force approach is that L is a huge number even for fairly small-sized problems and computationally it is not feasible to go through all of the L supports.

Orthogonal matching pursuit (OMP) [9, 10] is a greedy algorithm which overcomes this difficulty in a very simple iterative fashion. Instead of projecting $\mathbf{y}$ onto L different subspaces (brute-force approach), OMP makes projections onto n columns of $\mathbf{A}$. In comparison to L, n is a much smaller number and therefore computational complexity of OMP remains small. In each iteration, OMP projects the current residual error vector onto all n columns of $\mathbf{A}$ and selects that column which gives smallest ℓ_2-norm of the projection error. Residual error for next iteration is then computed by taking the difference between $\mathbf{y}$ and its orthogonal projection onto the subspace spanned by all the columns selected up to the current iteration. After k iterations OMP gives a set of k 'best' columns of $\mathbf{A}$. These k columns form an overdetermined system of equations and an estimate of $\mathbf{x}$ is obtained by solving that overdetermined system for observed vector $\mathbf{y}$. A formal definition of OMP is given in Algorithm 1.

In the above algorithm, the iteration number is denoted by a superscript in parenthesis, e.g., $S^{(i)}$ denotes the support set S at i-th iteration, while $\mathbf{a}_j$ denotes the j-th column of $\mathbf{A}$. In line 4, $\mathbf{r}$ denotes the residual error vector while $\left(\mathbf{a}_j^{\mathrm{T}}\mathbf{r}/\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j\right)\mathbf{a}_j$ in line 5 is the orthogonal projection of $\mathbf{r}$ onto $\mathbf{a}_j$ and therefore $\mathbf{r}-\left(\mathbf{a}_j^{\mathrm{T}}\mathbf{r}/\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j\right)\mathbf{a}_j$ is the projection error of $\mathbf{r}$ with $\mathbf{a}_j$. The solution of the optimization problem in line 7 can be obtained by solving an overdetermined linear system of equations. If $S^{(i)}$ indexed columns of $\mathbf{A}$ are linearly independent then the non-zero entries of $\widehat{\mathbf{x}}^{(i)}$ can be computed as linear least squares (LS) solution:

$$\widehat{\mathbf{x}}^{(i)}_{S^{(i)}} = \left(\mathbf{A}^T_{S^{(i)}}\mathbf{A}_{S^{(i)}}\right)^{-1}\mathbf{A}^T_{S^{(i)}}\mathbf{y},$$

Algorithm 1: Orthogonal Matching Pursuit (OMP) [9], [10]

Input: Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k
Initialize: $S^{(0)} \leftarrow \varnothing$, $\widehat{\mathbf{x}}^{(0)} \leftarrow \mathbf{0}$
for $i \leftarrow 1$ **to** k **do**
  $\mathbf{r} \leftarrow \mathbf{y} - \mathbf{A}\widehat{\mathbf{x}}^{(i-1)}$
  $\widehat{j} \leftarrow \underset{j \in [n]}{\operatorname{argmin}} \left\| \mathbf{r} - \left(\mathbf{a}_j^{\mathrm{T}} \mathbf{r} / \mathbf{a}_j^{\mathrm{T}} \mathbf{a}_j \right) \mathbf{a}_j \right\|_2^2$
  $S^{(i)} \leftarrow S^{(i-1)} \cup \left\{ \widehat{j} \right\}$
  $\widehat{\mathbf{x}}^{(i)} \leftarrow \underset{\widetilde{\mathbf{x}} \in \mathbb{R}^n}{\operatorname{argmin}} \ \|\mathbf{y} - \mathbf{A}\widetilde{\mathbf{x}}\|_2^2 \quad \text{s.t. } \operatorname{supp}(\widetilde{\mathbf{x}}) \subseteq S^{(i)}$
end
Output: Sparse vector $\mathbf{x}$ is estimated as $\widehat{\mathbf{x}} = \widehat{\mathbf{x}}^{(k)}$

where $\mathbf{A}_{S^{(i)}}$ is an $m \times i$ matrix that consists of only $S^{(i)}$ indexed columns of $\mathbf{A}$ and $\widehat{\mathbf{x}}^{(i)}_{S^{(i)}}$ is an $i \times 1$ vector that represents non-zero entries of $\widehat{\mathbf{x}}^{(i)}$.

OMP greedily builds up support of the unknown sparse vector one column at a time and thereby avoids large combinatorial computations of the brute-force method. There are also some theoretical results which give recovery guarantees of OMP for sparse signals. Although the performance guaranteed by these results is not as good as those for ℓ_1-based basis pursuit, they nevertheless provide necessary theoretical guarantees for OMP which is otherwise a somewhat heuristic method. One such result states that in the absence of measurement noise OMP will succeed in recovering an arbitrary k-sparse vector in k iterations with high probability if entries of the $m \times n$ measurement matrix are Gaussian or Bernoulli distributed and $m = \mathcal{O}(k \log n)$ [25]. This result does not mean that OMP will succeed in recovering all k-sparse vectors under the given conditions, i.e., if one tries to recover different k-sparse vectors one by one with the same measurement matrix then at some point OMP will fail to recover some particular k-sparse vector. In order to guarantee recovery of all k-sparse vectors, OMP needs to have more measurements, i.e. $m = \mathcal{O}(k^2 \log n)$ [26]. If OMP is allowed to run for more than k iterations then the number of measurements needed for guaranteed recovery can be further reduced from $\mathcal{O}(k^2 \log n)$ [27].

3.2 Compressive Sampling Matching Pursuit

Compressive sampling matching pursuit (CoSaMP) [12] is an iterative greedy algorithm for sparse signal recovery. The main theme of CoSaMP is somewhat similar to that of orthogonal matching pursuit (OMP), i.e., each iteration of CoSaMP involves both correlating the residual error vector with the columns of the measurement matrix and solving a least squares problem for the selected columns. CoSaMP also has a guaranteed upper bound on its error which makes it a much more reliable tool for sparse signal recovery.

A formal definition of CoSaMP is given in Algorithm 2. In each iteration of CoSaMP, one correlates the residual error vector $\mathbf{r}$ with the columns of the

Algorithm 2: Compressive Sampling Matching Pursuit (CoSaMP) [12]

1 **Input:** Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k
2 **Initialize:** $S^{(0)} \leftarrow \varnothing$, $\widehat{\mathbf{x}}^{(0)} \leftarrow \mathbf{0}$, $i \leftarrow 1$
3 **while** stopping criterion is not met **do**
4 $\quad \mathbf{r} \leftarrow \mathbf{y} - \mathbf{A}\widehat{\mathbf{x}}^{(i-1)}$
5 $\quad \widetilde{S}^{(i)} \leftarrow S^{(i-1)} \cup \operatorname{supp}\Big(H_{2k}(\mathbf{A}^{\mathrm{T}}\mathbf{r})\Big)$
6 $\quad \widetilde{\mathbf{x}}^{(i)} \leftarrow \underset{\mathbf{z} \in \mathbb{R}^n}{\operatorname{argmin}} \ \|\mathbf{y} - \mathbf{A}\mathbf{z}\|_2^2 \quad \text{s.t. } \operatorname{supp}(\mathbf{z}) \subseteq \widetilde{S}^{(i)}$
7 $\quad \widehat{\mathbf{x}}^{(i)} \leftarrow H_k(\widetilde{\mathbf{x}}^{(i)})$
8 $\quad S^{(i)} \leftarrow \operatorname{supp}(\widehat{\mathbf{x}}^{(i)})$
9 $\quad i \leftarrow i + 1$
10 **end**
11 **Output:** Sparse vector $\mathbf{x}$ is estimated as $\widehat{\mathbf{x}} = \widehat{\mathbf{x}}^{(i)}$ from the last iteration

measurement matrix $\mathbf{A}$. One then selects $2k$ columns of $\mathbf{A}$ corresponding to the $2k$ largest absolute values of the correlation vector $\mathbf{A}^{\mathrm{T}}\mathbf{r}$, where k denotes the sparsity of the unknown vector. This selection is represented by $\operatorname{supp}(H_{2k}(\mathbf{A}^{\mathrm{T}}\mathbf{r}))$ in line 5 of Algorithm 2. $H_{2k}(.)$ denotes the hard thresholding operator. Its output vector is obtained by setting all but the largest (in magnitude) $2k$ elements of its input vector to zero. The indices of the selected columns ($2k$ in total) are then added to the current estimate of the support (of cardinality k) of the unknown vector. A $3k$-sparse estimate of the unknown vector is then obtained by solving the least squares problem given in line 6 of Algorithm 2. The hard thresholding operator $H_k(.)$ is applied on the $3k$-sparse vector to obtain the k-sparse estimate of the unknown vector. This is a crucial step which ensures that the estimate of the unknown vector remains k-sparse. It also makes it possible to get rid of any columns that do not correspond to the true signal support but which might have been selected by mistake in the correlation step (line 5). This is a unique feature of CoSaMP which is missing in OMP. In OMP, if a mistake is made in selecting a column then the index of the selected column remains in the final estimate of the signal support and there is no way of removing it from the estimated support. In contrast, CoSaMP is more robust in dealing with the mistakes made in the estimation of the signal support.

The stopping criterion for CoSaMP can be set in a number of different ways. One way could be to run CoSaMP for a fixed number of iterations. Another possible way is to stop running CoSaMP once the ℓ_2-norm of the residual error vector $\mathbf{r}$ becomes smaller than a predefined threshold. The exact choice of a stopping criterion is often guided by practical considerations.

The computational complexity of CoSaMP is primarily dependent upon the complexity of solving the least squares problem in line 6 of Algorithm 2. The authors of CoSaMP suggest in [12] that the least squares problem should be solved iteratively using either Richardson's iteration [28, Section 7.2.3] or conjugate gradient [28, Section 7.4]. The authors of CoSaMP have also analyzed the error performance of these iterative methods in the context of CoSaMP. It turns out that the error decays

exponentially with each passing iteration, which means that in practice only a few iterations are enough to solve the least squares problem.

3.2.1 Error bound of CoSaMP

Let $\mathbf{x} \in \mathbb{R}^n$ be an arbitrary vector and $\mathbf{x}_k$ be the best k-sparse approximation of $\mathbf{x}$, i.e. amongst all k-sparse vectors, $\mathbf{x}_k$ is the one which is nearest (under any ℓ_p-metric) to $\mathbf{x}$. Furthermore let $\mathbf{x}$ be measured according to (2.2), i.e.,

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the measurement matrix and $\mathbf{v}$ is the noise vector. Now if the following condition holds,

- $\mathbf{A}$ satisfies RIP with $\delta_{4k} \leq 0.1$,

then CoSaMP is guaranteed to recover $\widehat{\mathbf{x}}^{(i)}$ at i-th iteration such that the error is bounded by [12]

$$\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2 \quad \leq \quad 2^{-i}\|\mathbf{x}\|_2 + 20\epsilon_k,$$

where

$$\epsilon_k \quad = \quad \|\mathbf{x} - \mathbf{x}_k\|_2 + \frac{1}{\sqrt{k}}\|\mathbf{x} - \mathbf{x}_k\|_1 + \|\mathbf{v}\|_2.$$

This means that the error term $2^{-i}\|\mathbf{x}\|_2$ decreases with each passing iteration of CoSaMP and the overall error $\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2$ is mostly determined by ϵ_k. In case $\mathbf{x}$ itself is k-sparse ($\mathbf{x}_k = \mathbf{x}$) then the error bound of CoSaMP is given by

$$\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2 \quad \leq \quad 2^{-i}\|\mathbf{x}\|_2 + 15\|\mathbf{v}\|_2.$$

3.3 Iterative Hard Thresholding

Consider again the optimization problem in (P2). Although the objective function is convex, the sparsity constraint is such that the feasible set is non-convex which makes the optimization problem hard to solve. Ignoring the sparsity constraint for a moment, one way of minimizing the objective function is to use gradient based methods. Iterative hard thresholding (IHT) [20, 11] takes such an approach. In each iteration of IHT the unknown sparse vector is estimated by taking a step in the direction of the negative gradient of the objective function. The estimate obtained this way will not be sparse in general. In order to satisfy the sparsity constraint IHT sets all but the largest (in magnitude) components of the estimated vector to zero using the so-called hard thresholding operator. A formal definition of IHT is given in Algorithm 3 while the objective function used in IHT and its negative gradient are given by

$$J(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2, \qquad -\nabla J(\mathbf{x}) = \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\mathbf{x}).$$

Algorithm 3: Iterative Hard Thresholding (IHT) [20], [11]

1 **Input:** Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k
2 **Initialize:** $\widehat{\mathbf{x}}^{(0)} \leftarrow \mathbf{0}$, $i \leftarrow 0$
3 **while** stopping criterion is not met **do**
4 $\quad \widetilde{\mathbf{x}}^{(i+1)} \leftarrow \widehat{\mathbf{x}}^{(i)} + \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\widehat{\mathbf{x}}^{(i)})$
5 $\quad \widehat{\mathbf{x}}^{(i+1)} \leftarrow H_k(\widetilde{\mathbf{x}}^{(i+1)})$
6 $\quad i \leftarrow i + 1$
7 **end**
8 **Output:** Sparse vector $\mathbf{x}$ is estimated as $\widehat{\mathbf{x}} = \widehat{\mathbf{x}}^{(i+1)}$ from the last iteration

In line 5 of the Algorithm 3, $H_k(.)$ denotes the hard thresholding operator. $H_k(\mathbf{x})$ is defined as the vector obtained by setting all but k largest (in magnitude) values of $\mathbf{x}$ to zero. Application of hard thresholding operator ensures that the sparsity constraint in (P2) is satisfied in every iteration of IHT. Lastly, IHT cannot run indefinitely so it needs a well defined stopping criterion. There are several possible ways in which a stopping criterion can be defined for IHT. For example, one could limit IHT to run for a fixed number of iterations, or another possibility is to terminate IHT if the estimate of the sparse vector does not change much from one iteration to the next. The choice of a particular stopping criterion is often guided by practical considerations.

Although IHT is seemingly based on an ad hoc construction, there are certain performance guarantees which make IHT a reliable tool for sparse signal recovery. Performance guarantees related to convergence and error bound of IHT are discussed next.

3.3.1 Convergence of IHT

Assuming $\mathbf{A} \in \mathbb{R}^{m \times n}$, IHT will converge to a local minimum of (P2) if the following conditions hold [20],

- $rank(\mathbf{A}) = m$,
- $\|\mathbf{A}\|_2 < 1$,

where $\|\mathbf{A}\|_2$ is the operator norm of $\mathbf{A}$ from ℓ_2 to ℓ_2 which is defined below:

$$\|\mathbf{A}\|_2 := \max_{\|\mathbf{x}\|_2 = 1} \|\mathbf{A}\mathbf{x}\|_2.$$

It can be shown that $\|\mathbf{A}\|_2$ is equivalent to the largest singular value of $\mathbf{A}$.

3.3.2 Error bound of IHT

Let $\mathbf{x} \in \mathbb{R}^n$ be an arbitrary vector and $\mathbf{x}_k$ be the best k-sparse approximation of $\mathbf{x}$. Furthermore let $\mathbf{x}$ be measured according to (2.2), i.e.,

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the measurement matrix and $\mathbf{v}$ is the noise vector. Now if the following condition holds,

- $\mathbf{A}$ satisfies RIP with $\delta_{3k} < \frac{1}{\sqrt{32}}$,

then IHT is guaranteed to recover $\widehat{\mathbf{x}}^{(i)}$ at i-th iteration such that the error is bounded by [11]

$$\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2 \quad \leq \quad 2^{-i}\|\mathbf{x}_k\|_2 + 6\epsilon_k,$$

where

$$\epsilon_k \quad = \quad \|\mathbf{x} - \mathbf{x}_k\|_2 + \frac{1}{\sqrt{k}}\|\mathbf{x} - \mathbf{x}_k\|_1 + \|\mathbf{v}\|_2.$$

This means that the error term $2^{-i}\|\mathbf{x}_k\|_2$ decreases with each passing iteration of IHT and the overall error $\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2$ is mostly determined by ϵ_k. In case $\mathbf{x}$ itself is k-sparse ($\mathbf{x}_k = \mathbf{x}$) then the error bound of IHT is given by

$$\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2 \quad \leq \quad 2^{-i}\|\mathbf{x}\|_2 + 5\|\mathbf{v}\|_2.$$

3.4 Normalized Iterative Hard Thresholding

The basic version of IHT discussed previously has nice performance guarantees which are valid as long as the underlying assumptions hold. If any one of the underlying assumptions does not hold then the performance of IHT degrades significantly. In particular, IHT is sensitive to scaling of the measurement matrix. This means that even though IHT may converge to a local minimum of (P2) with a measurement matrix satisfying $\|\mathbf{A}\|_2 < 1$, it may fail to converge for some scaling $c\mathbf{A}$ of the measurement matrix for which $\|c\mathbf{A}\|_2 \not< 1$. This is clearly an undesirable feature of IHT. In order to make IHT insensitive to operator norm of the measurement matrix, a slight modification to IHT has been proposed, called the normalized iterative hard thresholding (normalized IHT) [21]. The normalized IHT uses an adaptive step size (μ) in the gradient update equation of IHT. Any scaling of the measurement matrix is counterbalanced by an inverse scaling of the step size and as a result the algorithm remains stable. A suitable value of μ, which would ensure that the objective function decreases in each iteration, is then required to be recomputed in each iteration of the normalized IHT. The following discussion describes how μ is computed in the normalized IHT.

Let $\mathbf{g}$ denote the negative gradient of the objective function $J(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$, i.e.,

$$\mathbf{g} = -\nabla J(\mathbf{x}) = \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\mathbf{x}).$$

The gradient update equation for the normalized IHT in i-th iteration is then given by

$$\begin{aligned}\widetilde{\mathbf{x}}^{(i+1)} &= \widehat{\mathbf{x}}^{(i)} + \mu^{(i)}\mathbf{g}^{(i)} \\ &= \widehat{\mathbf{x}}^{(i)} + \mu^{(i)}\mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\widehat{\mathbf{x}}^{(i)}).\end{aligned}$$

An initial guess for the step size μ is obtained in a manner similar to the line search method. In the usual line search method, μ would be chosen such that it minimizes the objective function along the straight line in the direction of the negative gradient. For the given (quadratic) objective function this results in the following expression for optimal μ,

$$\begin{aligned}\mu^{(i)} &= \underset{\mu}{\arg\min}\ \left\|\mathbf{y} - \mathbf{A}\left(\widehat{\mathbf{x}}^{(i)} + \mu\mathbf{g}^{(i)}\right)\right\|_2^2 \\ &= \frac{{\mathbf{g}^{(i)}}^{\mathrm{T}}\mathbf{g}^{(i)}}{{\mathbf{g}^{(i)}}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{g}^{(i)}}.\end{aligned}$$

In case of the normalized IHT, the above value of μ is not guaranteed to decrease the objective function after the application of the hard thresholding operator ($H_k(.)$). To ensure that the objective function decreases even after the application of $H_k(.)$, μ is computed using a slightly modified version of the above formula, i.e., it includes the support of the current estimate of the sparse vector as given below:

$$\mu^{(i)} = \frac{{\mathbf{g}_{\Gamma^{(i)}}^{(i)}}^{\mathrm{T}}\mathbf{g}_{\Gamma^{(i)}}^{(i)}}{{\mathbf{g}_{\Gamma^{(i)}}^{(i)}}^{\mathrm{T}}\mathbf{A}_{\Gamma^{(i)}}^{\mathrm{T}}\mathbf{A}_{\Gamma^{(i)}}\mathbf{g}_{\Gamma^{(i)}}^{(i)}}, \tag{3.1}$$

where $\Gamma^{(i)} = \mathrm{supp}(\widehat{\mathbf{x}}^{(i)})$, $\mathbf{g}_{\Gamma^{(i)}}^{(i)}$ is the vector consisting of only those entries of $\mathbf{g}^{(i)}$ which are indexed by $\Gamma^{(i)}$, and $\mathbf{A}_{\Gamma^{(i)}}$ is the matrix consisting of only those columns of $\mathbf{A}$ which are indexed by $\Gamma^{(i)}$.

The value of μ computed in (3.1) is only an initial guess. In each iteration of the normalized IHT a move is made in the direction of the negative gradient using this initial guess and the estimate of the sparse vector is updated as

$$\begin{aligned}\widetilde{\mathbf{x}}^{(i+1)} &= \widehat{\mathbf{x}}^{(i)} + \mu^{(i)}\mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\widehat{\mathbf{x}}^{(i)}), \\ \widehat{\mathbf{x}}^{(i+1)} &= H_k(\widetilde{\mathbf{x}}^{(i+1)}).\end{aligned}$$

If the support of the new estimate $\widehat{\mathbf{x}}^{(i+1)}$ is same as that of the previous estimate $\widehat{\mathbf{x}}^{(i)}$ then the initial guess of μ is guaranteed to decrease the objective function. In this case the new estimate of the sparse vector is valid and the algorithm continues to next iteration. When supports of the two estimates are different then in order to guarantee a decrease in the objective function, μ needs to be compared with another parameter ω which is defined as,

$$\omega^{(i)} = (1-c)\frac{\|\widehat{\mathbf{x}}^{(i+1)} - \widehat{\mathbf{x}}^{(i)}\|_2^2}{\|\mathbf{A}(\widehat{\mathbf{x}}^{(i+1)} - \widehat{\mathbf{x}}^{(i)})\|_2^2},$$

where c is some small fixed constant. If μ is less than or equal to ω then μ is still guaranteed to decrease the objective function and there is no need to re-estimate the sparse vector. Otherwise μ is repeatedly scaled down and the sparse vector is re-estimated until μ becomes less than or equal to ω.

Just like in IHT, stopping criterion for the normalized IHT can be based on a fixed number of iterations or on the amount of change in the estimate of the sparse vector from one iteration to another. The stopping criterion can also be based on the norm of the residual error vector. The normalized IHT is formally defined in Algorithm 4.

Algorithm 4: Normalized Iterative Hard Thresholding (Normalized IHT) [21]

Input: Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k, small constant c, $\kappa > 1/(1-c)$

Initialize: $\widehat{\mathbf{x}}^{(0)} \leftarrow \mathbf{0}$, $\Gamma^{(0)} \leftarrow \mathrm{supp}(H_k(\mathbf{A}^{\mathrm{T}}\mathbf{y}))$, $i \leftarrow 0$

while stopping criterion is not met **do**

$\quad \mathbf{g}^{(i)} \leftarrow \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\widehat{\mathbf{x}}^{(i)})$

$\quad \mu^{(i)} \leftarrow \mathbf{g}_{\Gamma^{(i)}}^{(i)^{\mathrm{T}}} \mathbf{g}_{\Gamma^{(i)}}^{(i)} \Big/ \mathbf{g}_{\Gamma^{(i)}}^{(i)^{\mathrm{T}}} \mathbf{A}_{\Gamma^{(i)}}^{\mathrm{T}} \mathbf{A}_{\Gamma^{(i)}} \mathbf{g}_{\Gamma^{(i)}}^{(i)}$

$\quad \widetilde{\mathbf{x}}^{(i+1)} \leftarrow \widehat{\mathbf{x}}^{(i)} + \mu^{(i)}\mathbf{g}^{(i)}$

$\quad \widehat{\mathbf{x}}^{(i+1)} \leftarrow H_k(\widetilde{\mathbf{x}}^{(i+1)})$

$\quad \Gamma^{(i+1)} \leftarrow \mathrm{supp}(\widehat{\mathbf{x}}^{(i+1)})$

$\quad$ **if** $(\Gamma^{(i+1)} \neq \Gamma^{(i)})$ **then**

$\quad\quad \omega^{(i)} \leftarrow (1-c)\|\widehat{\mathbf{x}}^{(i+1)} - \widehat{\mathbf{x}}^{(i)}\|_2^2 \Big/ \|\mathbf{A}(\widehat{\mathbf{x}}^{(i+1)} - \widehat{\mathbf{x}}^{(i)})\|_2^2$

$\quad\quad$ **while** $(\mu^{(i)} > \omega^{(i)})$ **do**

$\quad\quad\quad \mu^{(i)} \leftarrow \mu^{(i)}/(\kappa(1-c))$

$\quad\quad\quad \widetilde{\mathbf{x}}^{(i+1)} \leftarrow \widehat{\mathbf{x}}^{(i)} + \mu^{(i)}\mathbf{g}^{(i)}$

$\quad\quad\quad \widehat{\mathbf{x}}^{(i+1)} \leftarrow H_k(\widetilde{\mathbf{x}}^{(i+1)})$

$\quad\quad\quad \omega^{(i)} \leftarrow (1-c)\|\widehat{\mathbf{x}}^{(i+1)} - \widehat{\mathbf{x}}^{(i)}\|_2^2 \Big/ \|\mathbf{A}(\widehat{\mathbf{x}}^{(i+1)} - \widehat{\mathbf{x}}^{(i)})\|_2^2$

$\quad\quad$ **end**

$\quad\quad \Gamma^{(i+1)} \leftarrow \mathrm{supp}(\widehat{\mathbf{x}}^{(i+1)})$

$\quad$ **end**

$\quad i \leftarrow i + 1$

end

Output: Sparse vector $\mathbf{x}$ is estimated as $\widehat{\mathbf{x}} = \widehat{\mathbf{x}}^{(i+1)}$ from the last iteration

3.4.1 Convergence of the normalized IHT

Assuming $\mathbf{A} \in \mathbb{R}^{m \times n}$, the normalized IHT will converge to a local minimum of (P2) if the following conditions hold [21],

- $rank(\mathbf{A}) = m$,
- $rank(\mathbf{A}_\Gamma) = k$ for all $\Gamma \subset [n]$ such that $|\Gamma| = k$ (which is another way of saying that $Spark(\mathbf{A})$ should be greater than k.)

For the convergence of the normalized IHT, $\|\mathbf{A}\|_2$ is no longer required to be less than one.

3.4.2 Error bound of the normalized IHT

Error bound of the normalized IHT is obtained using non-symmetric version of restricted isometry property. A matrix $\mathbf{A}$ satisfies non-symmetric RIP of order $2k$ if the following holds for some constants α_{2k}, β_{2k} and for all $2k$-sparse vectors $\mathbf{x}$,

$$\alpha_{2k}^2 \|\mathbf{x}\|_2^2 \leq \|\mathbf{A}\mathbf{x}\|_2^2 \leq \beta_{2k}^2 \|\mathbf{x}\|_2^2.$$

Let $\mathbf{x}$ be an arbitrary unknown vector which is measured under the model given in (2.2). Let $\mathbf{x}_k$ be the best k-sparse approximation of $\mathbf{x}$. If the normalized IHT computes the step size μ in every iteration using (3.1) then let $\gamma_{2k} = (\beta_{2k}^2/\alpha_{2k}^2) - 1$, otherwise let $\gamma_{2k} = \max\{1 - (\alpha_{2k}^2/(\kappa\beta_{2k}^2)), (\beta_{2k}^2/\alpha_{2k}^2) - 1\}$. Now if the following condition holds,

- $\mathbf{A}$ satisfies non-symmetric RIP with $\gamma_{2k} < 1/8$,

then the normalized IHT is guaranteed to recover $\widehat{\mathbf{x}}^{(i)}$ at i-th iteration such that the error is bounded by [21]

$$\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2 \quad \leq \quad 2^{-i}\|\mathbf{x}_k\|_2 + 8\epsilon_k,$$

where

$$\epsilon_k \quad = \quad \|\mathbf{x} - \mathbf{x}_k\|_2 + \frac{1}{\sqrt{k}}\|\mathbf{x} - \mathbf{x}_k\|_1 + \frac{1}{\beta_{2k}}\|\mathbf{v}\|_2.$$

Just as in IHT, the error term $2^{-i}\|\mathbf{x}_k\|_2$ decreases with each passing iteration and the overall error $\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2$ is mostly determined by ϵ_k. In the case when $\mathbf{x}$ itself is k-sparse ($\mathbf{x}_k = \mathbf{x}$) then the error bound of the normalized IHT is given by

$$\|\mathbf{x} - \widehat{\mathbf{x}}^{(i)}\|_2 \quad \leq \quad 2^{-i}\|\mathbf{x}\|_2 + \frac{8}{\beta_{2k}}\|\mathbf{v}\|_2.$$

Chapter 4

Bayesian Methods

The greedy algorithms described in Chapter 3 aim to recover the unknown sparse vector without making any prior assumptions about the probability distribution of the sparse vector. In the case when some prior knowledge about the distribution of the sparse vector is available, it would make sense to incorporate that prior knowledge into the estimation process. Bayesian methods, which view the unknown sparse vector as random, provide a systematic framework for doing that. By making use of Bayes' rule, these methods update the prior knowledge about the sparse vector in accordance with the new evidence or observations. This chapter deals with the methods that estimate the sparse vector in a Bayesian framework. As it turns out, computing the Bayesian minimum mean-squared error (MMSE) estimate of the sparse vector is infeasible due to the combinatorial complexity of the estimator. Therefore, methods which provide a good approximation to the MMSE estimator are discussed in this chapter.

4.1 Fast Bayesian Matching Pursuit

Fast Bayesian matching pursuit (FBMP) [22] is an algorithm that approximates the Bayesian minimum mean-squared error (MMSE) estimator of the sparse vector. FBMP assumes binomial prior on the signal sparsity and multivariate Gaussian prior on the noise. The non-zero values of the sparse vector are also assumed to have multivariate Gaussian prior. Since the exact MMSE estimator of the sparse vector requires combinatorially large number of computations, FBMP makes use of greedy iterations and derives a feasible approximation of the MMSE estimator. In this section we provide a detailed description of FBMP.

Let us consider the following linear model,

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v}, \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the unknown sparse vector which is now treated as a random vector. Let $\mathbf{s}$ be a binary vector whose entries are equal to one if the corresponding entries

of $\mathbf{x}$ are non-zero, and vice versa.

$$s_i = \begin{cases} 1 & \text{if } x_i \neq 0 \\ 0 & \text{if } x_i = 0 \end{cases} \qquad\qquad i = 1, 2, \ldots, n.$$

Since $\mathbf{x}$ is considered to be random, $\mathbf{s}$ will also be a random vector. By framing $\mathbf{x}$ and $\mathbf{s}$ as random vectors, available prior knowledge about these parameters can be incorporated into the model in the form of prior probability distributions. After observation vector $\mathbf{y}$ becomes available, uncertainty in the model is reduced by updating the distributions of $\mathbf{x}$ and $\mathbf{s}$. MMSE estimate is then simply the mean of the posterior distribution. The following discussion provides a concrete example on how Bayesian approach in FBMP is applied for estimating the sparse signal.

As usual, the measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is supposed to have fewer rows than columns ($m < n$). The components of the noise vector $\mathbf{v} \in \mathbb{R}^m$ are assumed to be independent and identically distributed (i.i.d.) Gaussian random variables having mean zero and variance σ_v^2. This implies that $\mathbf{v}$ has multivariate Gaussian distribution and the covariance matrix of $\mathbf{v}$ is a diagonal matrix with diagonal entries equal to σ_v^2,

$$\mathbf{v} \sim \mathcal{N}_m(\mathbf{0}, \sigma_v^2 \mathbf{I}).$$

The components s_i of $\mathbf{s}$ are assumed to be i.i.d. Bernoulli random variables with $\Pr(s_i = 1) = p_1$, therefore

$$p(s_i) = p_1^{s_i}(1 - p_1)^{1 - s_i},$$

$$p(\mathbf{s}) = \prod_{i=1}^{n} p(s_i) = p_1^{\|\mathbf{s}\|_0}(1 - p_1)^{n - \|\mathbf{s}\|_0}.$$

The vectors $\mathbf{x}$ and $\mathbf{s}$ are obviously not independent of each other and the distribution of x_i conditioned on s_i is assumed to be

$$x_i | (s_i = 1) \sim \mathcal{N}(0, \sigma_{x_i}^2),$$

$$x_i | (s_i = 0) = 0.$$

Let $\mathbf{x}_{\bar{\mathbf{s}}}$ be the vector consisting of those entries of $\mathbf{x}$ whose indices correspond to zero entries of $\mathbf{s}$ while $\mathbf{x}_\mathbf{s}$ denotes the vector consisting of those entries of $\mathbf{x}$ whose indices correspond to non-zero entries of $\mathbf{s}$. The vector $\mathbf{x}_{\bar{\mathbf{s}}} | \mathbf{s}$ is then by definition equal to $\mathbf{0}$, while assuming each x_i is independent of every other x_j and s_j ($j \neq i$), the distribution of $\mathbf{x}_\mathbf{s}$ conditioned on $\mathbf{s}$ is given by

$$\mathbf{x}_\mathbf{s} | \mathbf{s} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}(\mathbf{s})),$$

where $\mathbf{R}(\mathbf{s})$ is the diagonal covariance matrix whose diagonal entries are equal to $\sigma_{x_i}^2$. Since $\mathbf{x}_\mathbf{s} | \mathbf{s}$ and $\mathbf{v}$ have Gaussian distributions and they are assumed to be independent of each other, their joint distribution will also be multivariate Gaussian which is given by

$$\begin{bmatrix} \mathbf{x}_\mathbf{s} \\ \mathbf{v} \end{bmatrix} | \mathbf{s} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{R}(\mathbf{s}) & \mathbf{0} \\ \mathbf{0} & \sigma_v^2 \mathbf{I} \end{bmatrix} \right).$$

The joint vector formed from $\mathbf{y}$ and $\mathbf{x_s}$ can be written as

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{x_s}\end{bmatrix}|\mathbf{s} = \begin{bmatrix}\mathbf{A_s} & \mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x_s}\\ \mathbf{v}\end{bmatrix},$$

which is just a linear transformation of the joint vector formed from $\mathbf{x_s}$ and $\mathbf{v}$. Therefore, the joint distribution of $\mathbf{y}$ and $\mathbf{x_s}$ conditioned on $\mathbf{s}$ will also be multivariate Gaussian which is given by

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{x_s}\end{bmatrix}|\mathbf{s} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{A_s}\mathbf{R}(\mathbf{s})\mathbf{A_s}^{\mathrm{T}} + \sigma_v^2\mathbf{I} & \mathbf{A_s}\mathbf{R}(\mathbf{s})\\ \mathbf{R}(\mathbf{s})\mathbf{A_s}^{\mathrm{T}} & \mathbf{R}(\mathbf{s})\end{bmatrix}\right).$$

For notational convenience, let $\mathbf{\Phi}(\mathbf{s}) = \mathbf{A_s}\mathbf{R}(\mathbf{s})\mathbf{A_s}^{\mathrm{T}} + \sigma_v^2\mathbf{I}$. Since $[\mathbf{y}^{\mathrm{T}}\ \mathbf{x_s}^{\mathrm{T}}]^{\mathrm{T}}|\mathbf{s}$ is multivariate Gaussian, $\mathbf{x_s}|(\mathbf{y},\mathbf{s})$ will also be multivariate Gaussian and its mean is given by

$$\begin{aligned}\mathbb{E}[\mathbf{x_s}|(\mathbf{y},\mathbf{s})] &= \mathbb{E}[\mathbf{x_s}|\mathbf{s}] + \mathbf{Cov}(\mathbf{x_s},\mathbf{y}|\mathbf{s})\mathbf{Cov}(\mathbf{y}|\mathbf{s})^{-1}(\mathbf{y} - \mathbb{E}[\mathbf{y}|\mathbf{s}])\\ &= \mathbf{0} + \mathbf{R}(\mathbf{s})\mathbf{A_s}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{s})^{-1}(\mathbf{y} - \mathbf{0})\\ &= \mathbf{R}(\mathbf{s})\mathbf{A_s}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{s})^{-1}\mathbf{y},\end{aligned}$$

and

$$\mathbb{E}[\mathbf{x_{\bar{s}}}|(\mathbf{y},\mathbf{s})] = \mathbf{0}.$$

$\mathbb{E}[\mathbf{x}|(\mathbf{y},\mathbf{s})]$ is then obtained by merging together $\mathbb{E}[\mathbf{x_s}|(\mathbf{y},\mathbf{s})]$ and $\mathbb{E}[\mathbf{x_{\bar{s}}}|(\mathbf{y},\mathbf{s})]$. Finally, MMSE estimate of $\mathbf{x}|\mathbf{y}$, which is equal to the mean of the posterior distribution of $\mathbf{x}$, is given by

$$\widehat{\mathbf{x}}_{\mathrm{MMSE}} = \mathbb{E}[\mathbf{x}|\mathbf{y}] = \sum_{\mathbf{s}\in\mathcal{S}} p(\mathbf{s}|\mathbf{y})\mathbb{E}[\mathbf{x}|(\mathbf{y},\mathbf{s})], \tag{4.2}$$

where $\mathcal{S}$ denotes the set of all 2^n binary vectors of length n. From (4.2) it becomes evident that MMSE estimate of $\mathbf{x}$ is equal to the weighted sum of conditional expectations $\mathbb{E}[\mathbf{x}|(\mathbf{y},\mathbf{s})]$ with the weights given by the posterior distribution of $\mathbf{s}$. Since the summation in (4.2) needs to be evaluated over all $\mathcal{S}$ and $|\mathcal{S}| = 2^n$, it is not feasible to compute the MMSE estimate of $\mathbf{x}$. This motivates the development of approximate MMSE estimates that would not need to perform exponentially large number of computations. FBMP is an example of such a method.

The main idea in FBMP is to identify those binary vectors $\mathbf{s}$ that have high posterior probability mass $p(\mathbf{s}|\mathbf{y})$. These vectors would then be called dominant vectors since they have a larger influence on the accuracy of the MMSE approximation due to their larger weights. A set $\mathcal{S}_\star$ containing D number of dominant vectors can then be constructed. An approximate MMSE estimate of $\mathbf{x}$ can then be obtained by evaluating (4.2) over $\mathcal{S}_\star$ instead of $\mathcal{S}$,

$$\widehat{\mathbf{x}}_{\mathrm{AMMSE}} = \sum_{\mathbf{s}\in\mathcal{S}_\star} p(\mathbf{s}|\mathbf{y})\mathbb{E}[\mathbf{x}|(\mathbf{y},\mathbf{s})] \qquad \Big(|\mathcal{S}_\star| = D.\Big)$$

A central issue that needs to be solved is finding the D dominant vectors from $\mathcal{S}$. In principle, one could evaluate $p(\mathbf{s}|\mathbf{y})$ for all $\mathbf{s} \in \mathcal{S}$ and select D vectors with

largest values of $p(\mathbf{s}|\mathbf{y})$. But this is not feasible since the search space in this case would be exponentially large. FBMP overcomes this difficulty by using a greedy iterative approach. In FBMP, binary vectors $\mathbf{s}$ are selected on the basis of a metric $\mu(\mathbf{s})$ which is based on $p(\mathbf{s}|\mathbf{y})$.

$$\begin{aligned} p(\mathbf{s}|\mathbf{y}) &\propto p(\mathbf{s})p(\mathbf{y}|\mathbf{s}), \\ \mu(\mathbf{s}) := \log p(\mathbf{s})p(\mathbf{y}|\mathbf{s}) &= \log p(\mathbf{s}) + \log p(\mathbf{y}|\mathbf{s}), \end{aligned}$$

where

$$\begin{aligned} \log p(\mathbf{s}) &= \log \left(p_1^{\|\mathbf{s}\|_0} (1-p_1)^{n-\|\mathbf{s}\|_0} \right) \\ &= \|\mathbf{s}\|_0 \log p_1 + (n - \|\mathbf{s}\|_0) \log(1-p_1) \\ &= \|\mathbf{s}\|_0 \log(\tfrac{p_1}{1-p_1}) + n \log(1-p_1), \\ \log p(\mathbf{y}|\mathbf{s}) &= \log \left(\frac{1}{\sqrt{(2\pi)^m \det(\mathbf{\Phi}(\mathbf{s}))}} \exp(-\tfrac{1}{2}\mathbf{y}^{\mathrm{T}} \mathbf{\Phi}(\mathbf{s})^{-1} \mathbf{y}) \right) \\ &= -\frac{1}{2} \left(m \log 2\pi + \log \det(\mathbf{\Phi}(\mathbf{s})) + \mathbf{y}^{\mathrm{T}} \mathbf{\Phi}(\mathbf{s})^{-1} \mathbf{y} \right). \end{aligned}$$

Let $\mathcal{S}^{(1)}$ denote the set of all 1-sparse binary vectors of length n $\left(|\mathcal{S}^{(1)}| = n\right)$. In the first iteration of FBMP, $\mu(\mathbf{s})$ is computed for every $\mathbf{s} \in \mathcal{S}^{(1)}$ and then D vectors with largest values of $\mu(\mathbf{s})$ are selected. This set of selected vectors is denoted by $\mathcal{S}_\star^{(1)}$. For each $\mathbf{s} \in \mathcal{S}_\star^{(1)}$, there are $n-1$ vectors that are 2-sparse and whose support includes the support of $\mathbf{s}$. Let the set of all such 2-sparse vectors be denoted by $\mathcal{S}^{(2)}$ $\left(|\mathcal{S}^{(2)}| \leq D \cdot (n-1)\right)$. In the second iteration of FBMP, $\mu(\mathbf{s})$ is computed for every $\mathbf{s} \in \mathcal{S}^{(2)}$ and then the D vectors with largest values of $\mu(\mathbf{s})$ are selected. This set of selected vectors is then denoted by $\mathcal{S}_\star^{(2)}$. For each $\mathbf{s} \in \mathcal{S}_\star^{(2)}$, there are now $n-2$ vectors that are 3-sparse and whose support includes the support of $\mathbf{s}$. Let the set of all such 3-sparse vectors be denoted by $\mathcal{S}^{(3)}$ $\left(|\mathcal{S}^{(3)}| \leq D \cdot (n-2)\right)$. The third iteration and all the subsequent iterations proceed in a manner which is similar to the first two iterations, i.e., in each iteration i, $\mu(\mathbf{s})$ is computed for every $\mathbf{s} \in \mathcal{S}^{(i)}$ $\left(|\mathcal{S}^{(i)}| \leq D \cdot (n+1-i)\right)$ and then the D vectors with largest values of $\mu(\mathbf{s})$ are selected. This procedure continues for P number of iterations and the set of D dominant vectors is obtained from the final iteration, i.e., $\mathcal{S}_\star = \mathcal{S}_\star^{(P)}$. Since $\mathbf{s}$ is a binary vector, its ℓ_0-pseudonorm can be written as

$$\|\mathbf{s}\|_0 = \sum_{i=1}^{n} s_i.$$

Each s_i is an i.i.d. Bernoulli random variable with $\Pr(s_i = 1) = p_1$. Therefore, $\|\mathbf{s}\|_0$ will have binomial distribution with parameters n and p_1. The total number of iterations P in FBMP can be chosen such that $\Pr(\|\mathbf{s}\|_0 > P)$ remains sufficiently small.

An iterative method for selecting D dominant vectors out of 2^n binary vectors is described above. In each iteration of this method, one needs to compute $\mu(\mathbf{s})$ for each candidate vector $\mathbf{s}$, which requires computing the inverse of the matrix $\mathbf{\Phi}(\mathbf{s})$. Therefore, a naive implementation of FBMP would be computationally very inefficient. The paper on FBMP [22] describes an efficient implementation which improves the efficiency of FBMP by taking advantage of the dependencies between computations in successive iterations. The details of that implementation can be found in [22].

4.2 Randomized Orthogonal Matching Pursuit

Randomized orthogonal matching pursuit (RandOMP) [23], [29] is another algorithm for obtaining an approximate MMSE estimate of the sparse vector in the Bayesian linear model. In RandOMP, the total number of non-zero entries of the sparse vector is assumed to be fixed and known. RandOMP assumes multivariate Gaussian distributions for the non-zero entries of the sparse vector and the noise. RandOMP approximates the MMSE estimator using greedy iterations based on OMP.

The main theme of RandOMP is similar to that of FBMP but there exist some major differences. For example, instead of evaluating the sum in (4.2) over a small set of dominant vectors, RandOMP evaluates the sum over a small set of sample supports. These supports are randomly drawn from a distribution that closely approximates the posterior distribution of the signal support. Moreover, the approximation of the posterior distribution in RandOMP is based on a method which closely resembles OMP, whereas FBMP uses a completely different greedy method for approximating the posterior distribution. A detailed description of RandOMP follows next.

Let us consider the linear model of (4.1),

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^m$, and $m < n$. The vector $\mathbf{x}$ is again treated as a random vector whose support is defined as,

$$S = \{j : x_j \neq 0\},$$

where x_j indicates the j-th component of $\mathbf{x}$. The noise vector $\mathbf{v}$ is assumed to have zero mean multivariate Gaussian distribution with covariance matrix $\sigma_v^2\mathbf{I}$,

$$\mathbf{v} \sim \mathcal{N}_m(\mathbf{0}, \sigma_v^2\mathbf{I}).$$

It is further assumed that $\mathbf{x}$ is known to be k-sparse, i.e., $|S| = k$. Let Ω denote the set of all possible supports of $\mathbf{x}$ when $\mathbf{x}$ is k-sparse ($|\Omega| = \binom{n}{k}$). The vector S is assumed to have uniform prior distribution, i.e.,

$$p(S) = \begin{cases} \frac{1}{|\Omega|} & \text{if } S \in \Omega \\ 0 & \text{otherwise} \end{cases}.$$

MMSE estimate of $\mathbf{x}$ is then given by

$$\widehat{\mathbf{x}}_{\text{MMSE}} = \mathbb{E}[\mathbf{x}|\mathbf{y}] = \sum_{S\in\Omega} p(S|\mathbf{y})\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)], \tag{4.3}$$

where $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ is the MMSE estimate of $\mathbf{x}$ when both $\mathbf{y}$ and S are given.

$\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ can be derived in a manner similar to that of FBMP. Let $\mathbf{x}_S$ be a vector consisting of those components of $\mathbf{x}$ that are indexed by S. Then by definition, $\mathbf{x}_{\overline{S}}|S = \mathbf{0}$, where $\overline{S} = [n]\backslash S$ and $[n] = \{1, 2, \ldots, n\}$. The vector $\mathbf{x}_S$ is assumed to have zero mean multivariate Gaussian distribution with covariance matrix $\sigma_x^2\mathbf{I}$, i.e.,

$$\mathbf{x}_S|S \sim \mathcal{N}_k(\mathbf{0}, \sigma_x^2\mathbf{I}).$$

From Bayes' rule, the posterior distribution of $\mathbf{x}_S$ given $\mathbf{y}$ and S can be written as

$$p(\mathbf{x}_S|(\mathbf{y}, S)) = \frac{p(\mathbf{x}_S|S)p(\mathbf{y}|(\mathbf{x}_S, S))}{p(\mathbf{y}|S)},$$

where $p(\mathbf{y}|S)$ is a normalizing constant for fixed $\mathbf{y}$ and S, and

$$\begin{aligned} p(\mathbf{x}_S|S) &= \frac{1}{(2\pi)^{k/2}\sigma_x^k} \exp\left(-\frac{\mathbf{x}_S^{\mathrm{T}}\mathbf{x}_S}{2\sigma_x^2}\right), \\ p(\mathbf{y}|(\mathbf{x}_S, S)) &= \frac{1}{(2\pi)^{m/2}\sigma_v^m} \exp\left(-\frac{(\mathbf{y}-\mathbf{A}_S\mathbf{x}_S)^{\mathrm{T}}(\mathbf{y}-\mathbf{A}_S\mathbf{x}_S)}{2\sigma_v^2}\right). \end{aligned}$$

Therefore,

$$p(\mathbf{x}_S|(\mathbf{y}, S)) \propto \frac{1}{(2\pi)^{(k+m)/2}\sigma_x^k\sigma_v^m} \exp\left(-\frac{\mathbf{x}_S^{\mathrm{T}}\mathbf{x}_S}{2\sigma_x^2} - \frac{(\mathbf{y}-\mathbf{A}_S\mathbf{x}_S)^{\mathrm{T}}(\mathbf{y}-\mathbf{A}_S\mathbf{x}_S)}{2\sigma_v^2}\right).$$

Since both the prior $p(\mathbf{x}_S|S)$ and the likelihood $p(\mathbf{y}|(\mathbf{x}_S, S))$ are multivariate Gaussian with known covariance, the posterior $p(\mathbf{x}_S|(\mathbf{y}, S))$ will also be multivariate Gaussian which implies that the mean of the posterior distribution is equal to its mode. Therefore,

$$\begin{aligned} \mathbb{E}[\mathbf{x}_S|(\mathbf{y}, S)] &= \underset{\mathbf{x}_S}{\arg\max}\ \log p(\mathbf{x}_S|(\mathbf{y}, S)) \\ &= \underset{\mathbf{x}_S}{\arg\max}\ \left(-\frac{\mathbf{x}_S^{\mathrm{T}}\mathbf{x}_S}{2\sigma_x^2} - \frac{(\mathbf{y}-\mathbf{A}_S\mathbf{x}_S)^{\mathrm{T}}(\mathbf{y}-\mathbf{A}_S\mathbf{x}_S)}{2\sigma_v^2}\right). \end{aligned}$$

Setting the gradient of the objective function above to zero and solving for $\mathbf{x}_S$ gives

$$\mathbb{E}[\mathbf{x}_S|(\mathbf{y}, S)] = \left(\frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I}\right)^{-1} \frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{y}, \tag{4.4}$$

while

$$\mathbb{E}[\mathbf{x}_{\overline{S}}|(\mathbf{y}, S)] = \mathbf{0}. \tag{4.5}$$

For notational convenience, let

$$\mathbf{Q}_S = \frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I},$$

then $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ is obtained by merging $\mathbb{E}[\mathbf{x}_S|(\mathbf{y}, S)]$ with $\mathbb{E}[\mathbf{x}_{\overline{S}}|(\mathbf{y}, S)]$.

The posterior probability $p(S|\mathbf{y})$ used in (4.3) can also be derived using Bayes' rule, i.e.,

$$p(S|\mathbf{y}) = \frac{p(S)p(\mathbf{y}|S)}{p(\mathbf{y})}.$$

For a fixed $\mathbf{y}$, $p(\mathbf{y})$ is just a normalizing constant, while $p(S)$ is also constant for all $S \in \Omega$. Therefore, one can write

$$p(S|\mathbf{y}) \propto p(\mathbf{y}|S),$$

where $p(\mathbf{y}|S)$ is the likelihood function of S for a fixed $\mathbf{y}$. By marginalization over $\mathbf{x}_S$, $p(\mathbf{y}|S)$ can be written as

$$p(\mathbf{y}|S) \propto \int\limits_{\mathbf{x}_S \in \mathbb{R}^k} \exp\left(-\frac{\mathbf{x}_S^{\mathrm{T}}\mathbf{x}_S}{2\sigma_x^2} - \frac{(\mathbf{y} - \mathbf{A}_S\mathbf{x}_S)^{\mathrm{T}}(\mathbf{y} - \mathbf{A}_S\mathbf{x}_S)}{2\sigma_v^2}\right) \mathrm{d}\mathbf{x}_S.$$

After simplifying the above integral and dropping additional constant terms, one can write (cf. pages 214, 215 of [29])

$$p(\mathbf{y}|S) \propto q_S = \exp\left(\frac{\mathbf{y}^{\mathrm{T}}\mathbf{A}_S\mathbf{Q}_S^{-1}\mathbf{A}_S^{\mathrm{T}}\mathbf{y}}{2\sigma_v^4} + \frac{1}{2}\log\left(\det(\mathbf{Q}_S^{-1})\right)\right). \tag{4.6}$$

The posterior $p(S|\mathbf{y})$, which is proportional to the likelihood $p(\mathbf{y}|S)$, is then obtained by normalizing q_S, i.e.,

$$p(S|\mathbf{y}) = \frac{q_S}{\sum\limits_{\widetilde{S}\in\Omega} q_{\widetilde{S}}}.$$

In order to compute the MMSE estimate of $\mathbf{x}$, one needs to compute the summation in (4.3) for all $S \in \Omega$. This is not feasible since $|\Omega| = \binom{n}{k}$ is a huge number. In FBMP, the MMSE estimate was approximated by computing the summation over a small set of dominant vectors, i.e., vectors that had large posterior probability $p(S|\mathbf{y})$. RandOMP takes a different approach. The main idea in RandOMP is to draw L random supports from Ω according to the distribution $p(S|\mathbf{y})$. This set of L supports is denoted by $\Omega_\star$. The approximate MMSE estimate can then be computed by

$$\widehat{\mathbf{x}}_{\mathrm{AMMSE}} = \frac{1}{L}\sum_{S\in\Omega_\star} \mathbb{E}[\mathbf{x}|(\mathbf{y}, S)], \qquad \Big(|\Omega_\star| = L.\Big) \tag{4.7}$$

The weight $p(S|\mathbf{y})$ of each support S does not appear explicitly in (4.7). This is because the weight is implicitly applied during the process of random selection. The supports that have large weights are more likely to be selected than the supports that have small weights and thus one only needs to perform a simple average of the L terms.

Algorithm 5: Randomized Orthogonal Matching Pursuit (RandOMP) [29]

Input: Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k, number of draws L, data variance σ_x^2, noise variance σ_v^2

for $l \leftarrow 1$ **to** L **do**

  $S^{(0)} \leftarrow \varnothing$, $\mathbf{z}^{(0)} \leftarrow \mathbf{0}$

  for $i \leftarrow 1$ **to** k **do**

    $\mathbf{r} \leftarrow \mathbf{y} - \mathbf{A}\mathbf{z}^{(i-1)}$

    Draw an integer j randomly with probability proportional to

$$\tilde{q}_j = \exp\left(\frac{\sigma_x^2(\mathbf{a}_j^{\mathrm{T}}\mathbf{r})^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \sigma_v^2\right)} - \frac{1}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \frac{1}{\sigma_x^2}\right)\right)$$

    $S^{(i)} \leftarrow S^{(i-1)} \cup \{j\}$

    $\mathbf{z}^{(i)} \leftarrow \underset{\widetilde{\mathbf{z}} \in \mathbb{R}^n}{\arg\min} \ \|\mathbf{y} - \mathbf{A}\widetilde{\mathbf{z}}\|_2^2 \quad \text{s.t. } \mathrm{supp}(\widetilde{\mathbf{z}}) \subseteq S^{(i)}$

  end

  $S \leftarrow S^{(k)}$

  $\widehat{\mathbf{x}}_S^{(l)} \leftarrow \left(\frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I}\right)^{-1}\frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{y}, \qquad \widehat{\mathbf{x}}_{\overline{S}}^{(l)} \leftarrow \mathbf{0}$

end

$\widehat{\mathbf{x}} \leftarrow \frac{1}{L}\sum_{l=1}^{L}\widehat{\mathbf{x}}^{(l)}$

Output: $\widehat{\mathbf{x}}$ is the approximate MMSE estimate of $\mathbf{x}$

Although the summation in (4.7) involves a much smaller number of terms than the one in (4.3), it is still not feasible to compute (4.7) in its current form. This is due to the fact that $p(S|\mathbf{y})$ is a distribution of size $\binom{n}{k}$, and it is not feasible to draw random samples from such a large distribution. As in FBMP, one needs to form an approximation of this distribution and then draw random samples from that approximation. A brief description of the way RandOMP forms this approximation is given next.

Let us imagine that $\mathbf{x}$ was 1-sparse, i.e., $k = 1$. Now $|\Omega| = \binom{n}{1} = n$ and it becomes feasible to compute $p(S|\mathbf{y})$ for every $S \in \Omega$. Now q_S, as defined in (4.6), will be equal to

$$q_i = \exp\left(\frac{\sigma_x^2(\mathbf{a}_i^{\mathrm{T}}\mathbf{y})^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i + \sigma_v^2\right)} - \frac{1}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i + \frac{1}{\sigma_x^2}\right)\right), \tag{4.8}$$

where $i = 1, 2, \ldots, n$ and $\mathbf{a}_i$ denotes the i-th column of $\mathbf{A}$. The posterior probability of S is then given by

$$p(S = i|\mathbf{y}) = \frac{q_i}{\sum_{j=1}^{n} q_j}. \tag{4.9}$$

In the case when $k > 1$, RandOMP builds a random support iteratively by drawing one column in each iteration. In the first iteration, RandOMP draws a column according to the distribution in (4.9). For each subsequent iteration, RandOMP uses OMP algorithm to generate the residual error vector $\mathbf{r}$ (cf. line 4 in Algorithm 1). The weights q_i in (4.8) are then modified by replacing $\mathbf{y}$ with $\mathbf{r}$, i.e.,

$$\tilde{q}_i = \exp\left(\frac{\sigma_x^2(\mathbf{a}_i^{\mathrm{T}}\mathbf{r})^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i + \sigma_v^2\right)} - \frac{1}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_i^{\mathrm{T}}\mathbf{a}_i + \frac{1}{\sigma_x^2}\right)\right).$$

The modified weights are then used to draw columns in all of the subsequent iterations. After k iterations, RandOMP gives a randomly drawn support S of the sparse vector $\mathbf{x}$. $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ can be computed from (4.4) and (4.5). This process is repeated L number of times. Finally, the approximate MMSE estimate of $\mathbf{x}$ is obtained from (4.7). RandOMP is formally defined in Algorithm 5. It should be noted that the estimate $\widehat{\mathbf{x}}_{\mathrm{AMMSE}}$ of the sparse vector is itself not necessarily sparse. If a sparse estimate is desired then one can take the support of the k largest absolute entries of $\widehat{\mathbf{x}}_{\mathrm{AMMSE}}$ and evaluate $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ from (4.4) and (4.5) over this support. Such an estimate will obviously be k-sparse but it will likely have higher mean-squared error.

4.3 Randomized Iterative Hard Thresholding

Randomized iterative hard thresholding (RandIHT) [24] is another algorithm for approximating the MMSE estimator of the unknown sparse vector in the Bayesian linear model. RandIHT assumes the same signal model and prior distributions as those in RandOMP. Therefore, the posterior distribution of the signal support and the MMSE estimator have the same forms in both of these algorithms. The difference between the two algorithms lies in the manner in which the posterior distribution of the signal support is approximated. Whereas RandOMP uses OMP-based approximation, RandIHT approximates the posterior distribution of the signal support using IHT-based greedy iterations. A brief description of RandIHT is given below.

Assuming the same signal model and the prior distributions used previously in RandOMP, the posterior distribution of the support S of the sparse vector can be written as

$$p(S|\mathbf{y}) \propto \exp\left(\frac{\mathbf{y}^{\mathrm{T}}\mathbf{A}_S\mathbf{Q}_S^{-1}\mathbf{A}_S^{\mathrm{T}}\mathbf{y}}{2\sigma_v^4} + \frac{1}{2}\log\left(\det(\mathbf{Q}_S^{-1})\right)\right), \tag{4.10}$$

where $\mathbf{y}$ is the observed vector, $\mathbf{A}$ is the measurement matrix, $\mathbf{A}_S$ is the matrix consisting of those columns of $\mathbf{A}$ that are indexed by S, σ_v is the standard deviation of the measurement noise, $\mathbf{Q}_S$ is a matrix defined as

$$\mathbf{Q}_S = \frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I},$$

and σ_x is the standard deviation of the non-zero entries of the sparse vector $\mathbf{x}$. An

Algorithm 6: Randomized Iterative Hard Thresholding (RandIHT) [24]

1 **Input:** Measurement matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k, number of draws L, data variance σ_x^2, noise variance σ_v^2
2 **for** $l \leftarrow 1$ **to** L **do**
3 $\quad \acute{\mathbf{x}}^{(0)} \leftarrow \mathbf{0}$, $i \leftarrow 0$
4 $\quad$ **while** stopping criterion is not met **do**
5 $\quad\quad \widetilde{\mathbf{x}}^{(i+1)} \leftarrow \acute{\mathbf{x}}^{(i)} + \mathbf{A}^{\mathrm{T}}(\mathbf{y} - \mathbf{A}\acute{\mathbf{x}}^{(i)})$
6 $\quad\quad$ Draw a support S of cardinality k randomly using the weights $\tilde{q}_j$ $(j = 1, 2, \ldots, n)$ in the weighted random sampling algorithm

$$\tilde{q}_j = c \cdot \exp\left(\frac{\sigma_x^2(\widetilde{x}_j^{(i+1)})^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \sigma_v^2\right)} - \frac{1}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \frac{1}{\sigma_x^2}\right)\right)$$

7 $\quad\quad \acute{\mathbf{x}}_S^{(i+1)} \leftarrow \frac{1}{\sigma_v^2}\mathbf{Q}_S^{-1}\mathbf{A}_S^{\mathrm{T}}\mathbf{y}, \qquad \acute{\mathbf{x}}_{\overline{S}}^{(i+1)} \leftarrow \mathbf{0}$
8 $\quad\quad i \leftarrow i + 1$
9 $\quad$ **end**
10 $\quad \widehat{\mathbf{x}}^{(l)} \leftarrow \acute{\mathbf{x}}^{(i)}$
11 **end**
12 $\widehat{\mathbf{x}} \leftarrow \frac{1}{L}\sum_{l=1}^{L} \widehat{\mathbf{x}}^{(l)}$
13 **Output:** $\widehat{\mathbf{x}}$ is the approximate MMSE estimate of $\mathbf{x}$

approximate MMSE estimate of the unknown sparse vector can then be obtained as

$$\widehat{\mathbf{x}}_{\mathrm{AMMSE}} = \frac{1}{L}\sum_{S\in\Omega_\star} \mathbb{E}[\mathbf{x}|(\mathbf{y}, S)],$$

where $\Omega_\star$ is a set of L supports drawn randomly according to $p(S|\mathbf{y})$ and $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ can be computed in two parts, i.e.,

$$\mathbb{E}[\mathbf{x}_S|(\mathbf{y}, S)] = \frac{1}{\sigma_v^2}\mathbf{Q}_S^{-1}\mathbf{A}_S^{\mathrm{T}}\mathbf{y} \quad \text{and} \quad \mathbb{E}[\mathbf{x}_{\overline{S}}|(\mathbf{y}, S)] = \mathbf{0}. \tag{4.11}$$

Since drawing random samples directly from $p(S|\mathbf{y})$ is not feasible due to the large number of possible supports, one must draw samples from a small enough distribution that can closely approximate $p(S|\mathbf{y})$. In RandOMP, one uses the residual error vector $\mathbf{r}$ computed in line 4 of the Algorithm 1 to approximate the posterior distribution of the support S. In RandIHT, one instead uses the vector $\widetilde{\mathbf{x}}^{(i+1)}$ obtained in line 4 of the Algorithm 3 to form the following distribution:

$$\tilde{q}_j = c \cdot \exp\left(\frac{\sigma_x^2(\widetilde{x}_j^{(i+1)})^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \sigma_v^2\right)} - \frac{1}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \frac{1}{\sigma_x^2}\right)\right),$$

Algorithm 7: Weighted Random Sampling (WRS) [24]

1 **Input:** A vector of non-negative weights $\tilde{\mathbf{q}} \in \mathbb{R}^n$, cardinality k
2 **Initialize:** $S \leftarrow \varnothing$,
3 **for** $l \leftarrow 1$ **to** k **do**
4 Draw an integer $j \in \{1, 2, \ldots, n\} \backslash S$ randomly with probability

$$\frac{\tilde{q}_j}{\sum\limits_{i \in \{1,2,\ldots,n\} \backslash S} \tilde{q}_i}$$

5 $S \leftarrow S \cup \{j\}$
6 **end**
7 **Output:** S is the randomly drawn support having cardinality k

where $\tilde{x}_j^{(i+1)}$ is the j-th component of $\tilde{\mathbf{x}}^{(i+1)}$ and c is a normalizing constant which ensures that the above distribution sums up to one. In each iteration of RandIHT, one draws k samples without replacement from $\tilde{q}_j$ to form an estimate of the support S. One then computes the MMSE estimate $\acute{\mathbf{x}}$ of $\mathbf{x}$ for the support S in the current iteration using (4.11). These iterations continue until the stopping criterion is met, which can be based on a fixed number of iterations or on the norm of the residual error $\mathbf{y} - \mathbf{A}\acute{\mathbf{x}}$. The same procedure is repeated L number of times and the final estimate of the sparse vector is obtained from simple averaging of all L estimates. RandIHT is formally defined in Algorithm 6.

The support S drawn in each iteration of RandIHT has cardinality k. This is in contrast to RandOMP in which only one element of the support is drawn in each iteration. To draw the complete support with k elements, RandIHT uses weighted random sampling (WRS) algorithm defined in Algorithm 7. In WRS, one element of the support is drawn in each iteration and it takes k number of iterations to draw a complete support. Once an element is selected in an iteration, it must not be selected again in future iterations. This means that the elements are selected without replacement and therefore the weights $\tilde{q}_j$ need to be re-normalized in each iteration of WRS as shown in line 4 of Algorithm 7.

4.4 Randomized Compressive Sampling Matching Pursuit

The idea of randomizing greedy algorithms in order to find an approximate MMSE estimate of the sparse vector in the Bayesian linear model is considered again using the CoSaMP algorithm [12]. The algorithm thus obtained will be called randomized compressive sampling matching pursuit or RandCoSaMP for short. A brief description of RandCoSaMP is given below while a formal definition of RandCoSaMP is given in Algorithm 8.

We assume the same signal model utilized earlier in the case of RandOMP and

Algorithm 8: Randomized Compressive Sampling Matching Pursuit (RandCoSaMP)

Input: Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed vector $\mathbf{y} \in \mathbb{R}^m$, sparsity k, number of draws L, data variance σ_x^2, noise variance σ_v^2

for $l \leftarrow 1$ **to** L **do**

  $S^{(0)} \leftarrow \varnothing$, $\mathbf{z}^{(0)} \leftarrow \mathbf{0}$, $i \leftarrow 1$

  while stopping criterion is not met **do**

    $\mathbf{r} \leftarrow \mathbf{y} - \mathbf{A}\mathbf{z}^{(i-1)}$

    Draw a support $\widetilde{S}$ of cardinality $2k$ randomly using the weights $\tilde{q}_j$ $(j = 1, 2, \ldots, n)$ in the weighted random sampling algorithm

$$\tilde{q}_j = c \cdot \exp\left(\frac{\sigma_x^2 (\mathbf{a}_j^{\mathrm{T}} \mathbf{r})^2}{2\sigma_v^2 \left(\sigma_x^2 \mathbf{a}_j^{\mathrm{T}} \mathbf{a}_j + \sigma_v^2 \right)} - \frac{1}{2} \log\left(\frac{1}{\sigma_v^2} \mathbf{a}_j^{\mathrm{T}} \mathbf{a}_j + \frac{1}{\sigma_x^2} \right) \right)$$

    $S^{(i)} \leftarrow S^{(i-1)} \cup \widetilde{S}$

    $\widetilde{\mathbf{z}}_{S^{(i)}}^{(i)} \leftarrow \left(\frac{1}{\sigma_v^2} \mathbf{A}_{S^{(i)}}^{\mathrm{T}} \mathbf{A}_{S^{(i)}} + \frac{1}{\sigma_x^2} \mathbf{I} \right)^{-1} \frac{1}{\sigma_v^2} \mathbf{A}_{S^{(i)}}^{\mathrm{T}} \mathbf{y}, \qquad \widetilde{\mathbf{z}}_{\overline{S}^{(i)}}^{(i)} \leftarrow \mathbf{0}$

    $\mathbf{z}^{(i)} \leftarrow H_k(\widetilde{\mathbf{z}}^{(i)})$

    $S^{(i)} \leftarrow \mathrm{supp}(\mathbf{z}^{(i)})$

    $i \leftarrow i + 1$

  end

  $S \leftarrow S^{(i-1)}$

  $\widehat{\mathbf{x}}_S^{(l)} \leftarrow \left(\frac{1}{\sigma_v^2} \mathbf{A}_S^{\mathrm{T}} \mathbf{A}_S + \frac{1}{\sigma_x^2} \mathbf{I} \right)^{-1} \frac{1}{\sigma_v^2} \mathbf{A}_S^{\mathrm{T}} \mathbf{y}, \qquad \widehat{\mathbf{x}}_{\overline{S}}^{(l)} \leftarrow \mathbf{0}$

end

$\widehat{\mathbf{x}} \leftarrow \frac{1}{L} \sum_{l=1}^{L} \widehat{\mathbf{x}}^{(l)}$

Output: $\widehat{\mathbf{x}}$ is the approximate MMSE estimate of $\mathbf{x}$

RandIHT. This means that the posterior distribution $p(S|\mathbf{y})$ of the unknown signal support S is given by (4.10). Here we alter the iterations of CoSaMP to draw samples from an approximation of $p(S|\mathbf{y})$. As given in lines 4 and 5 of Algorithm 2, each iteration of CoSaMP involves correlating the current residual error vector $\mathbf{r}$ with the columns of the measurement matrix $\mathbf{A}$ and then selecting the indices of the $2k$ largest absolute entries of the correlation vector $\mathbf{A}^{\mathrm{T}}\mathbf{r}$. In RandCoSaMP, we replace this deterministic selection procedure with a randomized one. Instead of selecting indices of the $2k$ largest absolute entries of $\mathbf{A}^{\mathrm{T}}\mathbf{r}$, we select $2k$ indices randomly in accordance with the following distribution:

$$\tilde{q}_j = c \cdot \exp\left(\frac{\sigma_x^2 (\mathbf{a}_j^{\mathrm{T}} \mathbf{r})^2}{2\sigma_v^2 \left(\sigma_x^2 \mathbf{a}_j^{\mathrm{T}} \mathbf{a}_j + \sigma_v^2 \right)} - \frac{1}{2} \log\left(\frac{1}{\sigma_v^2} \mathbf{a}_j^{\mathrm{T}} \mathbf{a}_j + \frac{1}{\sigma_x^2} \right) \right),$$

where $\mathbf{a}_j$ is the j-th column of $\mathbf{A}$ and c is a normalizing constant. In this randomized selection procedure, an index j has a higher chance of selection if its corresponding entry $\mathbf{a}_j^{\mathrm{T}}\mathbf{r}$ has a large absolute value but there is still a chance that another index i, whose corresponding entry $\mathbf{a}_i^{\mathrm{T}}\mathbf{r}$ has a small absolute value, could be selected as well. This is in contrast with the deterministic selection procedure used in CoSaMP.

In order to ensure that the randomized selection in CoSaMP gives a set of $2k$ distinct indices, we draw each index randomly without replacement. This can be achieved using the WRS algorithm described in Algorithm 7. After selecting a support consisting of $2k$ distinct indices, RandCoSaMP performs the computations given in lines 7 - 11 of Algorithm 8. These are the same computations that are also performed in each iteration of CoSaMP, except for the line 8 where we have replaced the least squares solution with the Bayesian estimate. This process repeats and continues until the stopping criterion is met. At that point we would be successful in drawing a support S from a distribution which is an approximation to the posterior $p(S|\mathbf{y})$.

Once a support S is selected, RandCoSaMP computes $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ using (4.11). RandCoSaMP is run L number of times to obtain a set $\Omega_\star$ consisting of L randomly drawn supports. As before in RandOMP and RandIHT, the approximate MMSE estimate of the sparse vector is computed as

$$\widehat{\mathbf{x}}_{\mathrm{AMMSE}} = \frac{1}{L} \sum_{S \in \Omega_\star} \mathbb{E}[\mathbf{x}|(\mathbf{y}, S)].$$

The approximate MMSE estimate of the sparse vector is computed as an average of L different k-sparse estimates. In general, this estimate will not be k-sparse. If a k-sparse estimate is desired then one can take the support of the k largest absolute entries of $\widehat{\mathbf{x}}_{\mathrm{AMMSE}}$ and then compute $\mathbb{E}[\mathbf{x}|(\mathbf{y}, S)]$ over that support. Such an estimate will obviously be k-sparse and is called Sparse RandCoSaMP estimate.

4.5 Empirical Results

Now we give some empirical results comparing the performance of OMP, CoSaMP, normalized IHT, RandOMP, and RandCoSaMP. In these results, the length of the sparse vector is $n = 300$ and the number of measurements taken is $m = 150$. The $m \times n$ measurement matrix consists of i.i.d. Gaussian entries and each column of the measurement matrix is normalized to have unit norm. The locations of the non-zero entries of the sparse vector are selected uniformly at random whereas the values of the non-zero entries of the sparse vector are taken from the zero-mean Gaussian distribution with variance σ_x^2. The measurements are corrupted by an additive zero-mean Gaussian noise having variance σ_v^2. The signal-to-noise ratio (SNR) is defined as σ_x^2/σ_v^2 (or equivalently as $10 \log_{10}\left(\sigma_x^2/\sigma_v^2\right)$ in decibel). The approximate MMSE estimates in RandOMP and RandCoSaMP are obtained by averaging over $L = 10$ runs.

Figure 4.1 shows the fraction of the non-zero entries whose locations were correctly identified when the sparsity level k was 20. This fraction is considered as a function

of SNR. It can be seen that there is not much difference in the performance of the various algorithms used. Nevertheless, it can be said that CoSaMP seems to be the worst performing while Sparse RandOMP seems to perform best. Sparse RandCoSaMP is clearly performing better than CoSaMP. In Figure 4.2, we show the same performance measure but for sparsity level $k = 50$. Now the difference in performance of these algorithms is more noticeable and CoSaMP is lagging far behind the rest of the algorithms.

In Figures 4.3 and 4.4, we show mean-squared error between the true sparse vector and the estimates given by the algorithms considered in this study. As expected, Bayesian methods (RandOMP and RandCoSaMP) perform better than the greedy algorithms. Furthermore, RandOMP has a smaller mean-squared error and hence better performance than RandCoSaMP.

The obtained simulation results indicate that RandOMP is a better performing algorithm than RandCoSaMP. This is not surprising if we consider the fact that, under the given settings, the corresponding non-Bayesian greedy versions of these algorithms (i.e. OMP and CoSaMP) also show the similar difference in performance. Perhaps under different settings (e.g., different sparsity models as in [30]) CoSaMP can outperform OMP and then it would be interesting to see whether RandCoSaMP can do the same to RandOMP.

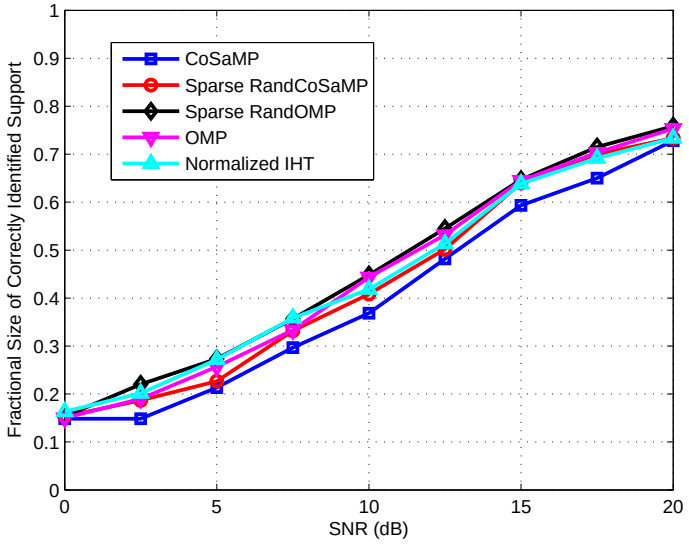


Figure 4.1: Fractional size of the correctly identified support vs SNR. The parameter values are: $k = 20$, $m = 150$, $n = 300$, $L = 10$. For the given settings, there is not much difference in the performance of these algorithms. However, Sparse RandOMP seems to be slightly better than the rest of the algorithms.

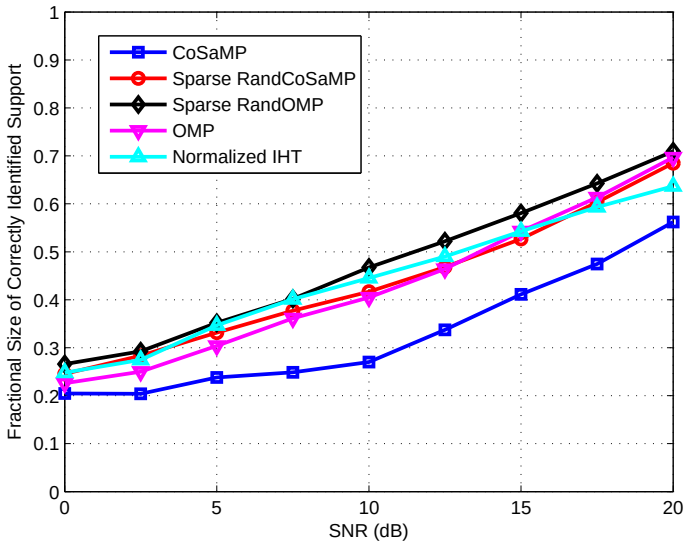


Figure 4.2: Fractional size of the correctly identified support vs SNR. The parameter values are: $k = 50$, $m = 150$, $n = 300$, $L = 10$. The difference in performance of these algorithms is more noticeable in this case. Sparse RandOMP is clearly the best performing algorithm. On the other hand, CoSaMP lags far behind the rest of the algorithms.

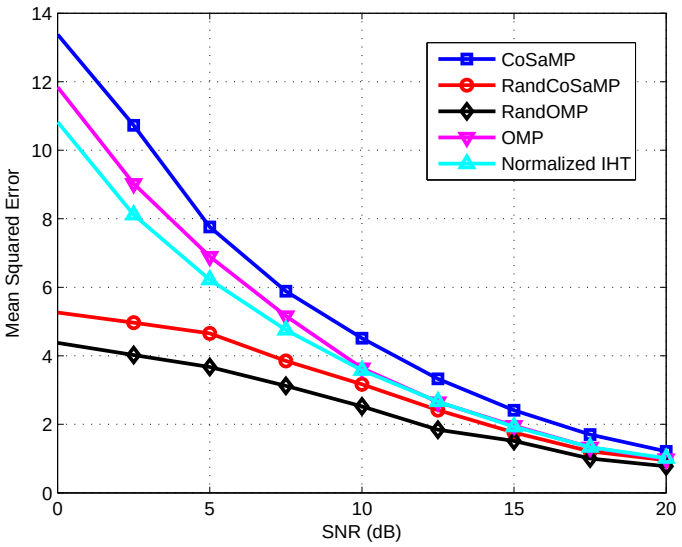


Figure 4.3: Mean-squared error vs SNR. The parameter values are: $k = 20$, $m = 150$, $n = 300$, $L = 10$. Bayesian algorithms (RandOMP and RandCoSaMP) have lower mean-squared error than non-Bayesian algorithms.

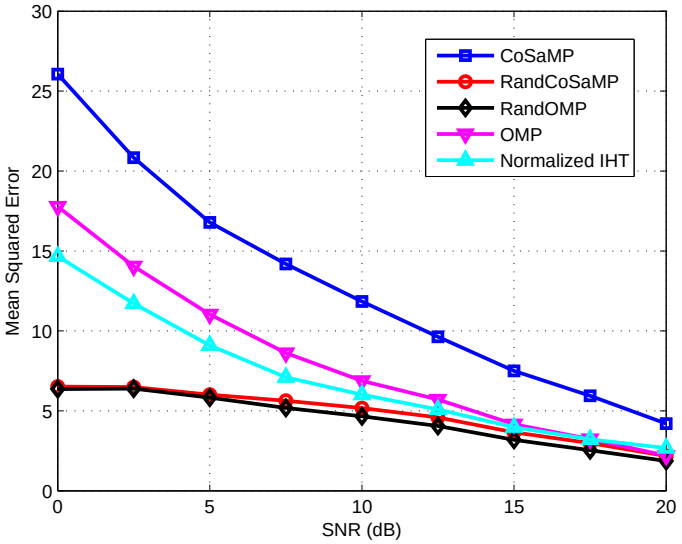


Figure 4.4: Mean-squared error vs SNR. The parameter values are: $k = 50$, $m = 150$, $n = 300$, $L = 10$. Bayesian algorithms (RandOMP and RandCoSaMP) have lower mean-squared error than non-Bayesian algorithms.

Chapter 5

Simultaneous Sparse Recovery

In this chapter we will discuss how one can recover multiple sparse vectors simultaneously from an underdetermined set of noisy linear measurements. In the first section of this chapter we formalize the signal model that will be used later for representing the underlying joint recovery problem. Next, we will discuss simultaneous orthogonal matching pursuit algorithm [31], a greedy algorithm used for the joint recovery of multiple sparse vectors. This will be followed by a discussion on randomized simultaneous orthogonal matching pursuit algorithm, a new algorithm developed in this thesis for approximating the joint Bayesian estimate of multiple sparse vectors. The algorithm will be generalized to recover complex-valued sparse vectors in the next chapter. Finally, we will give some empirical results comparing the performance of the greedy algorithm with the performance of its Bayesian counterpart.

5.1 Signal Model

We consider a problem in which the goal is to recover a set of q unknown sparse vectors $\mathbf{x}_i \in \mathbb{R}^n$ that are measured under the following linear model,

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{v}_i, \qquad i = 1, 2, \ldots, q. \tag{5.1}$$

The measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is fixed and is applied to all of the unknown sparse vectors. The observed vectors are denoted by $\mathbf{y}_i \in \mathbb{R}^m$, while the vectors $\mathbf{v}_i \in \mathbb{R}^m$ denote the unobservable measurement noise. It is further assumed that $m < n$, which means the unknown sparse vectors are measured in an underdetermined setting. The set of equations in (5.1) can be written in the following compact matrix form:

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{V}. \tag{5.2}$$

In the above equation, $\mathbf{Y} \in \mathbb{R}^{m \times q}$ is a matrix that holds each of the measured vectors $\mathbf{y}_i$ as one of its columns. Similarly, the columns of the matrix $\mathbf{X} \in \mathbb{R}^{n \times q}$ are formed by the unknown sparse vectors $\mathbf{x}_i$ while the columns of $\mathbf{V} \in \mathbb{R}^{m \times q}$ are formed by the noise vectors $\mathbf{v}_i$. The model given in (5.2) is commonly referred to as the multiple measurement vectors (MMV) model [32]. The task at hand is to recover the unknown

matrix $\mathbf{X}$ from the knowledge of just $\mathbf{A}$ and $\mathbf{Y}$. This problem is also referred to as multichannel sparse recovery problem [33].

If the unknown sparse vectors were totally independent of each other then there is no obvious gain in trying to recover these sparse vectors jointly. In such a case, one could simply solve the q equations in (5.1) one at a time using any one of the methods described in the previous chapters. Simultaneous sparse recovery makes sense only when different sparse vectors possess some common structure. For the given MMV model it is assumed that the support of each of the unknown sparse vectors is a subset of a common superset of cardinality $k < n$.

The row support of the matrix $\mathbf{X}$ is defined to be equal to the set of indices of the non-zero rows of $\mathbf{X}$. This is equal to the union of the supports of all the columns of $\mathbf{X}$, i.e.,

$$\begin{aligned} \text{rsupp}(\mathbf{X}) &= \bigcup_{j=1}^{q} \text{supp}(\mathbf{x}_j) \\ \text{supp}(\mathbf{x}_j) &= \{i : x_{ij} \neq 0\}, \end{aligned}$$

where x_{ij} denotes the i-th element of $\mathbf{x}_j$, or equivalently the element in the i-th row and j-th column of $\mathbf{X}$. The matrix $\mathbf{X}$ is said to be rowsparse with rowsparsity k when at most k rows of $\mathbf{X}$ contain non-zero entries.

The ℓ_0-pseudonorm of the rowsparse $\mathbf{X}$ is defined to be equal to the cardinality of its row support, i.e.,

$$\|\mathbf{X}\|_0 = |\text{rsupp}(\mathbf{X})|.$$

The Frobenius norm of $\mathbf{X}$ is defined as the square root of the sum of squares of the absolute values of the elements of $\mathbf{X}$, or equivalently as the square root of the sum of squared ℓ_2-norms of the columns of $\mathbf{X}$, i.e.,

$$\|\mathbf{X}\|_{\text{F}} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{q} |x_{ij}|^2} = \sqrt{\sum_{j=1}^{q} \|\mathbf{x}_j\|_2^2}.$$

5.2 Recovery Guarantees under the MMV Model

We assume that a rowsparse matrix $\mathbf{X}$ with rowsparsity k is measured with a measurement matrix $\mathbf{A}$ to produce the observed matrix $\mathbf{Y}$, i.e., $\mathbf{Y} = \mathbf{A}\mathbf{X}$. Then it is guaranteed that one can recover $\mathbf{X}$ from $\mathbf{Y}$ exactly if and only if [34]

$$Spark(\mathbf{A}) > 2k + 1 - rank(\mathbf{X}). \tag{5.3}$$

It was shown in Chapter 2 that $2 \leq Spark(\mathbf{A}) \leq m+1$, where m is the number of rows in $\mathbf{A}$. Since $\mathbf{X}$ has only k non-zero rows, we have $1 \leq rank(\mathbf{X}) \leq k$. From signal recovery perspective it is desired that $\mathbf{A}$ has maximum possible spark. Replacing $Spark(A)$ in (5.3) with its maximum possible value, we get

$$m > 2k - rank(\mathbf{X}). \tag{5.4}$$

If all the columns of $\mathbf{X}$ are multiples of each other then one does not expect their joint recovery to bring any improvement over individual recovery. This is indeed reflected in (5.4), where $rank(\mathbf{X}) = 1$ would result in the condition, $m \geq 2k$. This is the same necessary condition that we have for the single measurement vector (SMV) model as well. In the best case, when $rank(\mathbf{X}) = k$, we have

$$m \geq k + 1,$$

i.e., the minimum number of measurements required in the MMV model reduces to $k + 1$ as opposed to $2k$ in the SMV model.

5.3 Simultaneous Orthogonal Matching Pursuit

Simultaneous orthogonal matching pursuit (SOMP) [31] is an iterative greedy algorithm for the joint recovery of multiple sparse vectors under the MMV model. SOMP is based on an idea similar to the one used in OMP for the SMV model. In OMP, we iteratively project the residual error vector $\mathbf{r}$ onto each column of the measurement matrix $\mathbf{A}$ and then select the column that gives the smallest norm of the projection error. If each column of the measurement matrix has unit norm then one can also choose a column based on the absolute value of its inner product with the residual error vector. Such a choice would be valid because the column whose inner product with the residual error vector has the largest absolute value will be the one with the smallest norm of the projection error. This idea is extended in SOMP for the case of multiple sparse vectors.

A formal definition of SOMP is given in Algorithm 9. The signal model in (5.2) is assumed where the known matrices $\mathbf{A}$ and $\mathbf{Y}$ are given as input to SOMP. The columns of $\mathbf{A}$ are assumed to have unit norms. The unknown rowsparse matrix $\mathbf{X}$ is assumed to have known rowsparsity k. SOMP runs for k number of iterations (same as rowsparsity k). In each iteration, one first computes the current residual error matrix $\mathbf{R}$. This is shown in line 4 of Algorithm 9. In the next step, one takes each column of $\mathbf{A}$ and computes the sum of squares of its inner products with all the columns of $\mathbf{R}$. For the j-th column of $\mathbf{A}$, the sum is written as

$$\sum_{i=1}^{q} \left(\mathbf{r}_i^{\mathrm{T}} \mathbf{a}_j\right)^2,$$

where $\mathbf{r}_i$ denotes the i-th column of $\mathbf{R}$ while $\mathbf{a}_j$ denotes the j-th column of $\mathbf{A}$. The column of $\mathbf{A}$ that maximizes the above sum is then selected and its index is added to the list of selected columns. This is shown in lines 5 and 6 of Algorithm 9, where $[n]$ stands for the set $\{1, 2, \ldots, n\}$.

Lastly, one projects each column of the observed matrix $\mathbf{Y}$ onto the subspace spanned by the selected columns of $\mathbf{A}$. Each column of $\mathbf{X}$ is estimated by minimizing the ℓ_2-norm of the projection error of the corresponding column of $\mathbf{Y}$. This is shown in line 7 of Algorithm 9. If all the columns of $\mathbf{A}$ that are indexed by S are linearly independent then the optimization problem in line 7 has the following conventional

Algorithm 9: Simultaneous Orthogonal Matching Pursuit (SOMP) [31]

1 **Input:** Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ having columns of unit norm, observed matrix $\mathbf{Y} \in \mathbb{R}^{m \times q}$, rowsparsity k
2 **Initialize:** $S^{(0)} \leftarrow \varnothing$, $\widehat{\mathbf{X}}^{(0)} \leftarrow \mathbf{0}$
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\quad \mathbf{R} \leftarrow \mathbf{Y} - \mathbf{A}\widehat{\mathbf{X}}^{(i-1)}$
5 $\quad \widehat{j} \leftarrow \underset{j \in [n]}{\operatorname{argmax}} \left\| \mathbf{R}^T \mathbf{a}_j \right\|_2^2$
6 $\quad S^{(i)} \leftarrow S^{(i-1)} \cup \left\{ \widehat{j} \right\}$
7 $\quad \widehat{\mathbf{X}}^{(i)} \leftarrow \underset{\widetilde{\mathbf{X}} \in \mathbb{R}^{n \times q}}{\operatorname{argmin}} \ \|\mathbf{Y} - \mathbf{A}\widetilde{\mathbf{X}}\|_F^2 \quad \text{s.t. } \operatorname{rsupp}(\widetilde{\mathbf{X}}) \subseteq S^{(i)}$
8 **end**
9 **Output:** rowsparse matrix $\mathbf{X}$ is estimated as $\widehat{\mathbf{X}} = \widehat{\mathbf{X}}^{(k)}$

least squares solution:

$$\widehat{\mathbf{X}}_{(S)} = \left(\mathbf{A}_S^T \mathbf{A}_S \right)^{-1} \mathbf{A}_S^T \mathbf{Y}, \quad \widehat{\mathbf{X}}_{(\overline{S})} = \mathbf{0},$$

where $\widehat{\mathbf{X}}_{(S)}$ is the matrix formed by selecting those rows of $\widehat{\mathbf{X}}$ that are indexed by S, $\mathbf{A}_S$ is the matrix formed by selecting those columns of $\mathbf{A}$ that are indexed by S, and $\overline{S}$ is the complement of S (i.e. $\overline{S} = \{1, 2, \ldots, n\} \backslash S$). If the columns of $\mathbf{A}_S$ are linearly dependent then the optimization problem in line 7 does not have a unique solution. In this case one can compute the minimum Frobenius norm solution using the Moore-Penrose pseudoinverse of $\mathbf{A}_S$, i.e.,

$$\widehat{\mathbf{X}}_{(S)} = \mathbf{A}_S^+ \mathbf{Y}, \quad \widehat{\mathbf{X}}_{(\overline{S})} = \mathbf{0}.$$

The Moore-Penrose pseudoinverse $\mathbf{A}_S^+$ can be computed from the singular value decomposition (SVD) of $\mathbf{A}_S$.

The above procedure repeats for k iterations. In the end SOMP outputs an estimate of $\mathbf{X}$ containing at most k rows with non-zero entries.

5.4 Randomized Simultaneous Orthogonal Matching Pursuit

Here we propose a new algorithm that computes an approximate minimum mean-squared error (MMSE) estimate of the unknown rowsparse matrix in the MMV model. The algorithm is named randomized simultaneous orthogonal matching pursuit or RandSOMP for short. The setup used to derive RandSOMP is similar to the one used for randomized orthogonal matching pursuit (RandOMP) [23] except for the fact that RandSOMP deals with multiple sparse vectors. We will start by discussing the assumptions about the probability distributions of the signals in the MMV model. Then we will derive a closed-form expression of the MMSE

estimate of the rowsparse matrix. Since computing the exact MMSE estimate is again computationally infeasible, we will discuss how the greedy approach of SOMP can be used to approximate the MMSE estimate. Lastly, we will show how the approximate MMSE estimate, which itself is not rowsparse, is used to obtain a rowsparse estimate of the unknown matrix.

Let $\mathbf{X} \in \mathbb{R}^{n\times q}$ be an unknown rowsparse matrix having row support of known cardinality k. The matrix $\mathbf{X}$ is measured under the MMV model of (5.2), i.e.,

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{V},$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the known measurement matrix, $\mathbf{Y} \in \mathbb{R}^{m\times q}$ is the known matrix of observations, and $\mathbf{V} \in \mathbb{R}^{m\times q}$ represents the unknown measurement noise. Each element of the matrix $\mathbf{V}$ is supposed to be an independent and identically distributed (i.i.d.) normal random variable having mean zero and known variance σ_v^2. This means the matrix $\mathbf{V}$ has the following matrix variate normal distribution [35]:

$$\mathbf{V} \sim \mathcal{MN}_{m,q}\left(\mathbf{0}, \sigma_v^2\mathbf{I}_m, \mathbf{I}_q\right),$$

where $\mathbf{I}_m$ denotes the $m \times m$ identity matrix.

Definition 7. *A random matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ is said to have matrix variate normal distribution, indicated as $\mathbf{X} \sim \mathcal{MN}_{m,n}(\mathbf{M}, \mathbf{U}, \mathbf{V})$, if and only if*

$$\mathrm{vec}(\mathbf{X}) \sim \mathcal{N}_{mn}\left(\mathrm{vec}(\mathbf{M}), \mathbf{V} \otimes \mathbf{U}\right),$$

where $\mathbf{M} \in \mathbb{R}^{m\times n}$ is the mean matrix, $\mathbf{U} \in \mathcal{S}_{++}^m$ and $\mathbf{V} \in \mathcal{S}_{++}^n$ are the positive definite covariance matrices, vec(.) *is the vectorization operator, and $\otimes$ denotes the Kronecker product. The probability density of $\mathbf{X}$ is given by*

$$p(\mathbf{X}) = \frac{\exp\left(-\frac{1}{2}\mathrm{Tr}\left(\mathbf{V}^{-1}\left(\mathbf{X}-\mathbf{M}\right)^{\mathrm{T}}\mathbf{U}^{-1}\left(\mathbf{X}-\mathbf{M}\right)\right)\right)}{(2\pi)^{mn/2}|\mathbf{V}|^{m/2}|\mathbf{U}|^{n/2}},$$

where Tr(.) *denotes the trace operator. $\mathbf{X}$ also satisfies the following stochastic decomposition:*

$$\mathbf{X} = \mathbf{M} + \mathbf{U}^{1/2}\mathbf{X}_0\mathbf{V}^{1/2},$$

where $\mathbf{X}_0 \in \mathbb{R}^{m\times n}$ is a random matrix whose each entry is independent and identically distributed standard normal random variable.

Let S denote the row support of $\mathbf{X}$. Since the aim of RandSOMP is to estimate $\mathbf{X}$ in the Bayesian framework, it is essential to treat $\mathbf{X}$ and its row support S as random variables having known prior probability distributions. As mentioned earlier, the row support S is assumed to have fixed known cardinality k. Therefore, the total number of different possible row supports of $\mathbf{X}$ is $\binom{n}{k}$. Let Ω denote the set of all such row supports, with cardinality $|\Omega| = \binom{n}{k}$. It is further assumed that, within the set Ω, S has uniform prior distribution, i.e.,

$$p(S) = \begin{cases} \frac{1}{|\Omega|} & \text{if } S \in \Omega \\ 0 & \text{otherwise} \end{cases}.$$

For a given row support S, let $\mathbf{X}_{(S)}|S$ denote the $k \times q$ matrix formed by collecting all the non-zero rows of $\mathbf{X}$, i.e. the rows that are indexed by S. Then $\mathbf{X}_{(\overline{S})}|S = \mathbf{0}$ by definition, where $\overline{S}$ is the complement of S (i.e. $\overline{S} = \{1, 2, \ldots, n\} \backslash S$). Furthermore, let each element of $\mathbf{X}_{(S)}|S$ be an i.i.d. normal random variable having mean zero and known variance σ_x^2. This means $\mathbf{X}_{(S)}|S$ has the following matrix variate normal distribution:

$$\mathbf{X}_{(S)}|S \sim \mathcal{MN}_{k,q}\left(\mathbf{0}, \sigma_x^2 \mathbf{I}_k, \mathbf{I}_q\right).$$

The MMSE estimate of $\mathbf{X}$ is obtained as the mean of the posterior distribution of $\mathbf{X}$, i.e.,

$$\widehat{\mathbf{X}}_{\text{MMSE}} = \mathbb{E}[\mathbf{X}|\mathbf{Y}] = \sum_{S \in \Omega} p(S|\mathbf{Y}) \mathbb{E}[\mathbf{X}|(\mathbf{Y}, S)]. \tag{5.5}$$

The matrix $\mathbb{E}[\mathbf{X}|(\mathbf{Y}, S)]$ in (5.5) can be evaluated as two separate parts, i.e., as $\mathbb{E}[\mathbf{X}_{(\overline{S})}|(\mathbf{Y}, S)]$ and $\mathbb{E}[\mathbf{X}_{(S)}|(\mathbf{Y}, S)]$. The sub-matrix $\mathbb{E}[\mathbf{X}_{(\overline{S})}|(\mathbf{Y}, S)]$ is equal to $\mathbf{0}$ by definition. In order to evaluate the sub-matrix $\mathbb{E}[\mathbf{X}_{(S)}|(\mathbf{Y}, S)]$ one can use Bayes' rule as follows,

$$p(\mathbf{X}_{(S)}|(\mathbf{Y}, S)) = \frac{p(\mathbf{X}_{(S)}|S) p(\mathbf{Y}|(\mathbf{X}_{(S)}, S))}{p(\mathbf{Y}|S)}, \tag{5.6}$$

where $p(\mathbf{Y}|S)$ is a normalizing constant for fixed $\mathbf{Y}$ and S, and

$$p(\mathbf{X}_{(S)}|S) = \frac{1}{(2\pi\sigma_x^2)^{kq/2}} \exp\left(-\frac{1}{2}\text{Tr}\left(\frac{\mathbf{X}_{(S)}^{\text{T}} \mathbf{X}_{(S)}}{\sigma_x^2}\right)\right).$$

The trace is defined for a square matrix and is equal to the sum of the diagonal entries of the matrix. Moreover,

$$\mathbf{Y}|(\mathbf{X}_{(S)}, S) = \mathbf{A}_S \mathbf{X}_{(S)} + \mathbf{V},$$

where $\mathbf{A}_S$ is the matrix obtained by taking those columns of $\mathbf{A}$ that are indexed by S. For a given $\mathbf{X}_{(S)}$, $\mathbf{A}_S \mathbf{X}_{(S)}$ is a constant. Therefore,

$$\mathbf{Y}|(\mathbf{X}_{(S)}, S) \sim \mathcal{MN}_{m,q}\left(\mathbf{A}_S \mathbf{X}_{(S)}, \sigma_v^2 \mathbf{I}_m, \mathbf{I}_q\right),$$

and

$$p(\mathbf{Y}|(\mathbf{X}_{(S)}, S)) = \frac{1}{(2\pi\sigma_v^2)^{mq/2}} \exp\left(-\frac{1}{2}\text{Tr}\left(\frac{\left(\mathbf{Y} - \mathbf{A}_S \mathbf{X}_{(S)}\right)^{\text{T}} \left(\mathbf{Y} - \mathbf{A}_S \mathbf{X}_{(S)}\right)}{\sigma_v^2}\right)\right).$$

By using the closed-form expressions of $p(\mathbf{X}_{(S)}|S)$ and $p(\mathbf{Y}|(\mathbf{X}_{(S)}, S))$, and ignoring the normalizing constant $p(\mathbf{Y}|S)$, we can rewrite (5.6) as

$$p(\mathbf{X}_{(S)}|(\mathbf{Y}, S)) \propto \exp\left(-\frac{1}{2\sigma_x^2}\left\|\mathbf{X}_{(S)}\right\|_F^2 - \frac{1}{2\sigma_v^2}\left\|\mathbf{Y} - \mathbf{A}_S \mathbf{X}_{(S)}\right\|_F^2\right).$$

Since both the prior $p(\mathbf{X}_{(S)}|S)$ and the likelihood $p(\mathbf{Y}|(\mathbf{X}_{(S)}, S))$ have the form of a matrix variate normal distribution with known variance, the posterior $p(\mathbf{X}_{(S)}|(\mathbf{Y}, S))$

will also have the form of a matrix variate normal distribution. Therefore, the mean of the posterior will be equal to its mode, i.e.,

$$\begin{aligned}\mathbb{E}[\mathbf{X}_{(S)}|(\mathbf{Y},S)] &= \underset{\mathbf{X}_{(S)}\in\mathbb{R}^{k\times q}}{\operatorname{argmax}}\ p(\mathbf{X}_{(S)}|(\mathbf{Y},S)) \\ &= \underset{\mathbf{X}_{(S)}\in\mathbb{R}^{k\times q}}{\operatorname{argmax}}\ \log(p(\mathbf{X}_{(S)}|(\mathbf{Y},S))) \\ &= \underset{\mathbf{X}_{(S)}\in\mathbb{R}^{k\times q}}{\operatorname{argmax}}\ -\frac{1}{2\sigma_x^2}\left\|\mathbf{X}_{(S)}\right\|_F^2 - \frac{1}{2\sigma_v^2}\left\|\mathbf{Y}-\mathbf{A}_S\mathbf{X}_{(S)}\right\|_F^2.\end{aligned}$$

The above optimization problem can be solved by setting the matrix gradient of the objective function to $\mathbf{0}$ and solving the resulting set of equations. This results in the following closed-form solution:

$$\mathbb{E}[\mathbf{X}_{(S)}|(\mathbf{Y},S)] = \left(\frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I}_k\right)^{-1}\frac{1}{\sigma_v^2}\mathbf{A}_S^{\mathrm{T}}\mathbf{Y}. \tag{5.7}$$

In order to derive $p(S|\mathbf{Y})$ in (5.5), one can again use Bayes' rule,

$$p(S|\mathbf{Y}) = \frac{p(S)p(\mathbf{Y}|S)}{p(\mathbf{Y})},$$

where $p(\mathbf{Y})$ is a normalizing constant for a fixed $\mathbf{Y}$, and $p(S)$ is also constant for all $S\in\Omega$. Therefore,

$$p(S|\mathbf{Y}) \propto p(\mathbf{Y}|S).$$

Moreover,

$$\begin{aligned}p(\mathbf{Y}|S) &= \int_{\mathbf{X}_{(S)}\in\mathbb{R}^{k\times q}} p\Big((\mathbf{Y},\mathbf{X}_{(S)})|S\Big)\mathrm{d}\mathbf{X}_{(S)} \\ &= \int_{\mathbf{X}_{(S)}\in\mathbb{R}^{k\times q}} p(\mathbf{X}_{(S)}|S)p\Big(\mathbf{Y}|(\mathbf{X}_{(S)},S)\Big)\mathrm{d}\mathbf{X}_{(S)} \\ &\propto \int_{\mathbf{X}_{(S)}\in\mathbb{R}^{k\times q}} \exp\left(-\frac{1}{2\sigma_x^2}\left\|\mathbf{X}_{(S)}\right\|_F^2 - \frac{1}{2\sigma_v^2}\left\|\mathbf{Y}-\mathbf{A}_S\mathbf{X}_{(S)}\right\|_F^2\right)\mathrm{d}\mathbf{X}_{(S)} \\ &= \int_{\mathrm{vec}(\mathbf{X}_{(S)})\in\mathbb{R}^{kq}} \exp\left(-\frac{\left\|\mathrm{vec}(\mathbf{X}_{(S)})\right\|_2^2}{2\sigma_x^2} - \frac{\left\|\mathrm{vec}(\mathbf{Y}-\mathbf{A}_S\mathbf{X}_{(S)})\right\|_2^2}{2\sigma_v^2}\right)\mathrm{d}\,\mathrm{vec}(\mathbf{X}_{(S)}).\end{aligned}$$

Recognizing the fact that

$$\mathrm{vec}(\mathbf{Y}-\mathbf{A}_S\mathbf{X}_{(S)}) = \mathrm{vec}(\mathbf{Y}) - (\mathbf{I}_q\otimes\mathbf{A}_S)\mathrm{vec}(\mathbf{X}_{(S)}),$$

the above integral can be simplified as (cf. pages 214, 215 of [29])

$$p(\mathbf{Y}|S) \propto w_S = \exp\left(\frac{\mathrm{vec}(\mathbf{Y})^{\mathrm{T}}(\mathbf{I}_q \otimes \mathbf{A}_S)\mathbf{Q}_S^{-1}(\mathbf{I}_q \otimes \mathbf{A}_S)^{\mathrm{T}}\mathrm{vec}(\mathbf{Y})}{2\sigma_v^4} \dots \right.$$
$$\left. \dots + \frac{1}{2}\log\left(\det(\mathbf{Q}_S^{-1})\right)\right)$$
$$= \exp\left(\frac{\mathrm{vec}(\mathbf{A}_S^{\mathrm{T}}\mathbf{Y})^{\mathrm{T}}\mathbf{Q}_S^{-1}\mathrm{vec}(\mathbf{A}_S^{\mathrm{T}}\mathbf{Y})}{2\sigma_v^4} + \frac{1}{2}\log\left(\det(\mathbf{Q}_S^{-1})\right)\right),$$

where

$$\mathbf{Q}_S = \frac{1}{\sigma_v^2}\mathbf{I}_q \otimes \mathbf{A}_S^{\mathrm{T}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I}_{kq}.$$

Finally, the posterior $p(S|\mathbf{Y})$ of S, which is proportional to its likelihood $p(\mathbf{Y}|S)$, is obtained by normalizing w_S, i.e.,

$$p(S|\mathbf{Y}) = \frac{w_S}{\sum\limits_{\widetilde{S}\in\Omega} w_{\widetilde{S}}}. \tag{5.8}$$

The MMSE estimate of $\mathbf{X}$ given in (5.5) can be expressed in closed-form using (5.7) and (5.8). But the summation in (5.5) needs to be evaluated for all possible $S \in \Omega$, which is computationally infeasible due to the large size of Ω $\left(|\Omega| = \binom{n}{k}\right)$. One could approximate the MMSE estimate by randomly drawing a small number of sample row supports according to $p(S|\mathbf{Y})$ and evaluating the summation in (5.5) over these row supports. Again, due to the large size of the sample space Ω, this approach is not feasible. Next we describe how one can overcome these limitations via SOMP-like greedy pursuit. We propose to draw row supports iteratively by sampling one element at a time from a much smaller space. This results in row supports that are drawn from an approximation of $p(S|\mathbf{Y})$.

Suppose the rowsparsity of $\mathbf{X}$ is one, i.e. $|S| = k = 1$. This implies

$$\Omega = \{1, 2, \dots, n\},$$
$$\mathbf{Q}_{S=j} = \left(\frac{1}{\sigma_v^2}\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \frac{1}{\sigma_x^2}\right)\mathbf{I}_q \qquad j \in \Omega,$$

and

$$p(S=j|\mathbf{Y}) \propto w_{S=j} = \exp\left(\frac{\mathbf{a}_j^{\mathrm{T}}\mathbf{Y}\mathbf{Q}_{S=j}^{-1}\mathbf{Y}^{\mathrm{T}}\mathbf{a}_j}{2\sigma_v^4} + \frac{1}{2}\log\left(\det(\mathbf{Q}_{S=j}^{-1})\right)\right)$$
$$= \exp\left(\frac{\sigma_x^2\|\mathbf{Y}^{\mathrm{T}}\mathbf{a}_j\|_2^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \sigma_v^2\right)} - \frac{q}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \frac{1}{\sigma_x^2}\right)\right), \tag{5.9}$$

where $\mathbf{a}_j$ is the j-th column of $\mathbf{A}$. Since the size of the sample space ($|\Omega|$) is now equal to n, drawing a random row support S becomes trivial. For the case when $|S| = k > 1$, we propose an iterative greedy process that looks similar to SOMP and

Algorithm 10: Randomized Simultaneous Orthogonal Matching Pursuit (Rand-SOMP)

Input: Measurement matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, observed matrix $\mathbf{Y} \in \mathbb{R}^{m \times q}$, rowsparsity k, number of draws L, data variance σ_x^2, noise variance σ_v^2

for $l \leftarrow 1$ **to** L **do**

  $S^{(0)} \leftarrow \varnothing$, $\widetilde{\mathbf{X}}^{(0)} \leftarrow \mathbf{0}$

  for $i \leftarrow 1$ **to** k **do**

    $\mathbf{R} \leftarrow \mathbf{Y} - \mathbf{A}\widetilde{\mathbf{X}}^{(i-1)}$

    Draw an integer j randomly with probability proportional to

$$w_{S=j} = \exp\left(\frac{\sigma_x^2 \|\mathbf{R}^{\mathrm{T}}\mathbf{a}_j\|_2^2}{2\sigma_v^2\left(\sigma_x^2\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \sigma_v^2\right)} - \frac{q}{2}\log\left(\frac{1}{\sigma_v^2}\mathbf{a}_j^{\mathrm{T}}\mathbf{a}_j + \frac{1}{\sigma_x^2}\right)\right)$$

    $S^{(i)} \leftarrow S^{(i-1)} \cup \{j\}$

    $\widetilde{\mathbf{X}}^{(i)}_{(S^{(i)})} \leftarrow \left(\frac{1}{\sigma_v^2}\mathbf{A}_{S^{(i)}}^{\mathrm{T}}\mathbf{A}_{S^{(i)}} + \frac{1}{\sigma_x^2}\mathbf{I}_i\right)^{-1}\frac{1}{\sigma_v^2}\mathbf{A}_{S^{(i)}}^{\mathrm{T}}\mathbf{Y}, \qquad \widetilde{\mathbf{X}}^{(i)}_{(\overline{S}^{(i)})} \leftarrow \mathbf{0}$

  end

  $\widehat{\mathbf{X}}^{(l)} \leftarrow \widetilde{\mathbf{X}}^{(k)}$

end

$\widehat{\mathbf{X}} \leftarrow \frac{1}{L}\sum_{l=1}^{L}\widehat{\mathbf{X}}^{(l)}$

Output: $\widehat{\mathbf{X}}$ is the approximate MMSE estimate of $\mathbf{X}$

which builds up the random row support one element at a time (For the reference see Algorithm 10). In the first iteration, we randomly draw an element j_1 of S with probability proportional to the weights given in (5.9). We compute the MMSE estimate $\widetilde{\mathbf{X}}$ of $\mathbf{X}$ assuming $S = \{j_1\}$ and also compute the residual error matrix $\mathbf{R} = \mathbf{Y} - \mathbf{A}\widetilde{\mathbf{X}}$. In the second iteration we modify the probability weights in (5.9) by replacing the matrix $\mathbf{Y}$ with $\mathbf{R}$. We randomly draw another element j_2 of the row support using the modified weights and compute the MMSE estimate assuming $S = \{j_1, j_2\}$. This process continues for k iterations. At the end of k iterations we have a randomly drawn row support $S = \{j_1, j_2, \ldots, j_k\}$ from a distribution that approximates $p(S|\mathbf{Y})$.

To reduce the approximation error in the MMSE estimate, we run the above greedy process L number of times. Let $\Omega_\star$ denote the set of L row supports obtained from the L independent runs. The approximate MMSE estimate of $\mathbf{X}$ is then given by

$$\widehat{\mathbf{X}}_{\mathrm{AMMSE}} = \frac{1}{L}\sum_{S \in \Omega_\star}\mathbb{E}[\mathbf{X}|(\mathbf{Y}, S)]. \tag{5.10}$$

The posterior probability mass $p(S|\mathbf{Y})$ of each row support does not appear explicitly in the above formula. This is because the row supports having high probability are more likely to be selected in the sampling process than the row supports with low

probability. This means the row supports are represented in $\Omega_\star$ in (approximate) proportion to their probability masses. Hence, to approximate (5.5) simple averaging suffices.

In general, the approximate MMSE estimate of $\mathbf{X}$ given in (5.10) will not be rowsparse. In order to obtain an estimate which is not only rowsparse but is also related to the approximate MMSE estimate, we propose an approach similar to the one in RandOMP. Let $H_k(\widehat{\mathbf{X}}_{\text{AMMSE}})$ denote the matrix obtained by setting all but the k largest (in ℓ_2-norm) rows of $\widehat{\mathbf{X}}_{\text{AMMSE}}$ to $\mathbf{0}$. Let $\widetilde{S}$ be the row support of $H_k(\widehat{\mathbf{X}}_{\text{AMMSE}})$. A rowsparse estimate of $\mathbf{X}$ is then obtained as

$$\begin{aligned} \widetilde{S} &= \text{rsupp}\Big(H_k(\widehat{\mathbf{X}}_{\text{AMMSE}})\Big), \\ \widehat{\mathbf{X}}_{\text{rowsparse}} &= \mathbb{E}[\mathbf{X}|(\mathbf{Y}, \widetilde{S})]. \end{aligned} \tag{5.11}$$

In order to distinguish between the two estimates given in (5.10) and (5.11), we will refer to the latter as Sparse RandSOMP estimate.

5.5 Empirical Results

Now we present some empirical results comparing the performance of SOMP and RandSOMP. We also include the results of the case where RandOMP is used to recover the individual sparse vectors one by one. It will become evident that, for the MMV model, joint recovery of the rowsparse matrix makes more sense than recovering the individual sparse vectors separately. RandSOMP performs better than RandOMP both in terms of mean-squared error (MSE) and probability of exact support recovery. RandSOMP also outperforms SOMP in terms of MSE, which is not surprising since RandSOMP is meant to approximate the MMSE estimate. More importantly, RandSOMP also outperforms SOMP in terms of probability of exact support recovery.

The simulation setting used to generate the results is described next. We generate $q = 30$ sparse vectors each of length $n = 300$. All the sparse vectors have the same randomly chosen support of cardinality $k = 20$. All non-zero entries of the sparse vectors are independently drawn from the standard normal distribution (i.e., $\sigma_x^2 = 1$). Each sparse vector is measured using a 150×300 measurement matrix (i.e., $m = 150$). Each entry of the measurement matrix is independently drawn from the standard normal distribution and each column is normalized to have unit norm. The measurements are contaminated by zero mean additive Gaussian noise with variance σ_v^2. The signal-to-noise ratio (SNR) is defined as the ratio σ_x^2/σ_v^2 (or equivalently as $10\log_{10}\Big(\sigma_x^2/\sigma_v^2\Big)$ in decibel). The number of randomly drawn row supports in both RandSOMP and RandOMP is $L = 10$. The experiments are repeated 100 times to average out the noise in the results.

In Figure 5.1, we plot the probability of exact support recovery versus different values of SNR. Since RandSOMP and RandOMP produce estimates that are not necessarily rowsparse, we use the sparsified variants of these estimates to compute the probability of exact support recovery. As one can see, Sparse RandSOMP outperforms both SOMP and Sparse RandOMP.

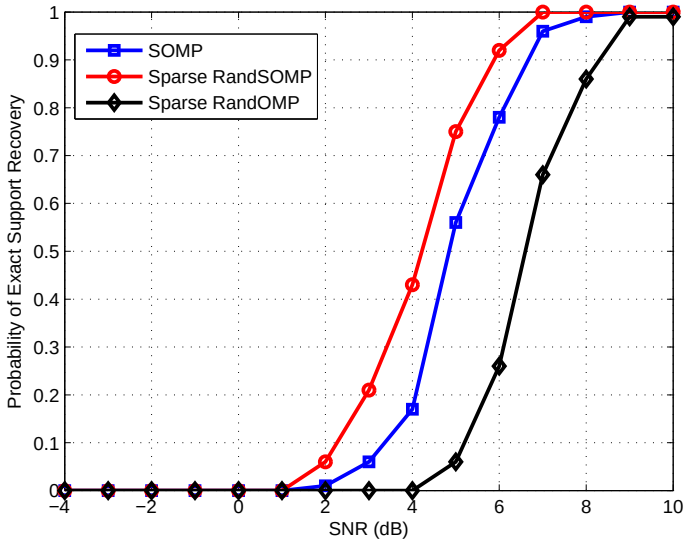


Figure 5.1: Probability of exact support recovery vs SNR. The parameter values are: $m = 150$, $n = 300$, $k = 20$, $L = 10$, $q = 30$. Sparse RandSOMP provides higher probability of exact support recovery than both SOMP and Sparse RandOMP.

In Figure 5.2, we plot the relative MSE of the three algorithms under consideration. At low values of SNR, RandSOMP and RandOMP show similar performance while the error is much larger for SOMP. This is expected since both RandSOMP and RandOMP approximate the MMSE estimate. With increasing SNR, the performance of RandOMP improves much more slowly in comparison to RandSOMP. On the other hand, the performance of SOMP improves much more sharply and at 10 dB SNR SOMP performs almost as good as RandSOMP.

In another simulation example, we fix the SNR value at 5 dB and vary the number of measurement vectors q. We keep the values of other parameters as before. Figure 5.3 shows the probability of exact support recovery in this case. When $q = 1$, the MMV model is reduced to the SMV model and thus SOMP and Sparse RandSOMP become equivalent to OMP and Sparse RandOMP respectively. As q increases we expect to gain some improvement through joint processing of multiple measurement vectors. This is evident from Figure 5.3. At $q = 1$, the probability of exact support recovery is almost 0 for both SOMP and Sparse RandSOMP. With increasing q the probability of exact support recovery increases such that at $q = 51$ it is almost 1 for both SOMP and Sparse RandSOMP. Furthermore, for all the given values of q, Sparse RandSOMP performs better than SOMP.

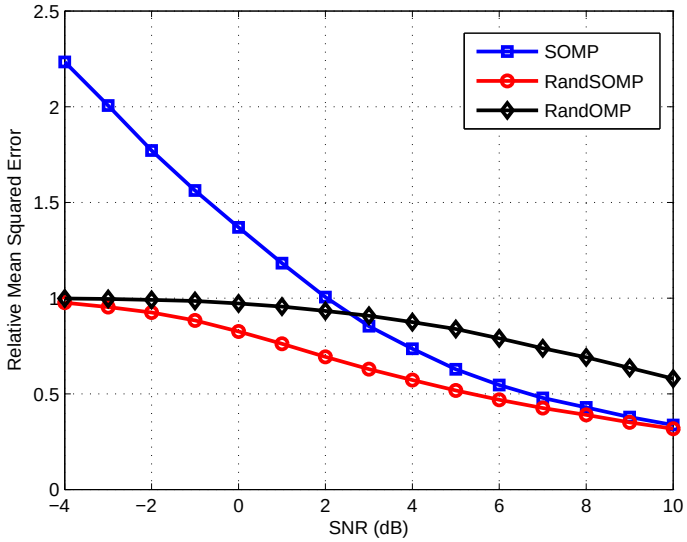


Figure 5.2: Relative mean-squared error vs SNR. The parameter values are: $m = 150$, $n = 300$, $k = 20$, $L = 10$, $q = 30$. RandSOMP provides lower MSE than both SOMP and RandOMP.

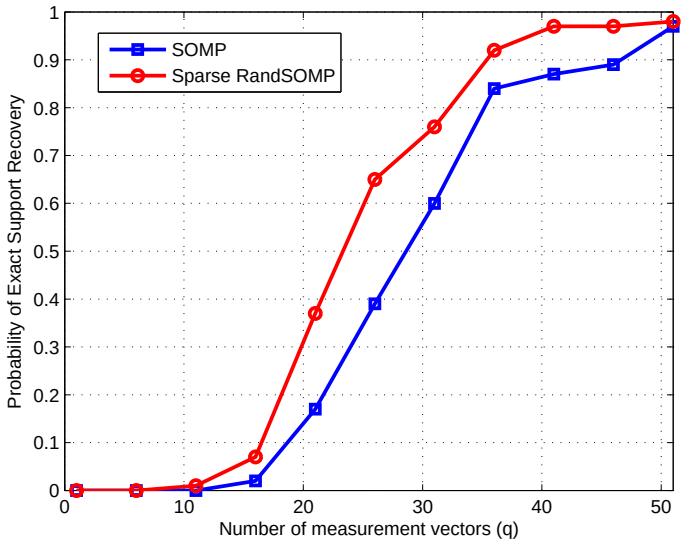


Figure 5.3: Probability of exact support recovery vs number of measurement vectors (q). The parameter values are: $m = 150$, $n = 300$, $k = 20$, $L = 10$, SNR $= 5$dB. Sparse RandSOMP provides higher probability of exact support recovery than SOMP.

Chapter 6

RandSOMP: The Complex-valued Case

In Chapter 5, we developed a novel Bayesian algorithm called RandSOMP to recover real-valued multichannel sparse vectors. In many practical applications, one needs to recover signals that are complex-valued. In this Chapter we will generalize the RandSOMP algorithm to deal with complex-valued data. We will assume that the non-zero entries of the multichannel sparse vectors and the noise have complex normal prior distribution. We will derive a closed-form solution of the MMSE estimator of the multichannel sparse vectors. As before, the MMSE estimator is infeasible to compute due to its combinatorial complexity. We will then show how SOMP-like greedy iterations can be used to approximate the MMSE estimator. We will evaluate the performance of the generalized RandSOMP algorithm using simulated data and demonstrate its advantage over SOMP in terms of higher probability of exact support recovery and lower mean-squared error.

We consider the multiple measurement vectors (MMV) model [32] in which the goal is to recover a set of q unknown complex-valued signal vectors $\mathbf{x}_i \in \mathbb{C}^n$ from a set of q measurement vectors $\mathbf{y}_i \in \mathbb{C}^m$. The vectors $\mathbf{x}_i$ are assumed to be sparse such that the cardinality of the union of their supports is k. The sparse vectors are measured according to the model, $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{v}_i$, $i = 1, 2, \ldots, q$ where $\mathbf{A} \in \mathbb{C}^{m \times n}$ is a known measurement matrix with $m < n$ and $\mathbf{v}_i \in \mathbb{C}^m$ denote the unobservable measurement *noise*. In matrix form, the model can be written as

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{V}, \tag{6.1}$$

where $\mathbf{Y} \in \mathbb{C}^{m \times q}$ with $\mathbf{y}_i$ as its ith column vector, $\mathbf{X} \in \mathbb{C}^{n \times q}$ contains the unknown sparse vectors $\mathbf{x}_i$ while $\mathbf{V} \in \mathbb{C}^{m \times q}$ is the noise matrix.

6.1 The MMSE Estimator

6.1.1 Assumptions

The elements of the noise matrix $\mathbf{V}$ are assumed to be independent and identically distributed (i.i.d.) complex normal random variables with zero mean and known

variance σ_v^2, so $\mathbf{V}$ has *matrix variate complex normal (MCN) distribution*, denoted as $\mathbf{V} \sim \mathcal{MCN}_{m,q}(\mathbf{0}, \sigma_v^2\mathbf{I}_m, \mathbf{I}_q)$, where $\mathbf{I}_m$ denotes the $m \times m$ identity matrix. In the Bayesian framework we treat $\mathbf{X}$ and its row support $S = \text{rsupp}(\mathbf{X})$ as random variables with known prior probability distributions. The row support S has known fixed cardinality k (i.e. $|S| = k$) and a uniform prior distribution (i.e., all row supports are equiprobable), $p(S) = 1/|\Omega|$ for $S \in \Omega$, where Ω denotes the set of all row supports that have cardinality k and $|\Omega| = \binom{n}{k}$. For a given row support S, let $\mathbf{X}_{(S)}$ denote the $k \times q$ matrix restricted to those k rows of $\mathbf{X}$ that are indexed by the support set S. Then $\mathbf{X}_{(\overline{S})} = \mathbf{0}$ by definition, where $\overline{S}$ is the complement of S. Furthermore, let each element of $\mathbf{X}_{(S)}$ be an i.i.d. complex normal random variable with zero mean and known variance σ_x^2, so $\mathbf{X}_{(S)} \sim \mathcal{MCN}_{k,q}(\mathbf{0}, \sigma_x^2\mathbf{I}_k, \mathbf{I}_q)$, with p.d.f.

$$p(\mathbf{X}_{(S)} \,|\, S) = \frac{1}{(\pi\sigma_x^2)^{kq}} \exp\left(-\frac{1}{\sigma_x^2}\left\|\mathbf{X}_{(S)}\right\|_F^2\right), \tag{6.2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, i.e., $\|\mathbf{X}\|_F = \sqrt{\text{Tr}(\mathbf{X}^{\text{H}}\mathbf{X})} = \|\text{vec}(\mathbf{X})\|$, where $\text{vec}(\mathbf{X})$ is a vector formed by stacking the columns of $\mathbf{X}$ on top of each other and $\|\cdot\|$ denotes the usual Euclidean norm. We define the signal-to-noise ratio (SNR) as $\gamma = \sigma_x^2/\sigma_v^2$.

6.1.2 Derivation of the MMSE estimator

The MMSE estimate of $\mathbf{X}$ is obtained as the conditional mean, i.e., the mean of the posterior distribution of $\mathbf{X}$,

$$\widehat{\mathbf{X}}_{\text{MMSE}} = \mathbb{E}\left[\mathbf{X} \,|\, \mathbf{Y}\right] = \sum_{S\in\Omega} p(S|\mathbf{Y})\mathbb{E}\left[\mathbf{X} \,|\mathbf{Y}, S\right], \tag{6.3}$$

where $\mathbb{E}[\mathbf{X}|\mathbf{Y}, S]$ is k-rowsparse since $\mathbb{E}\Big[\mathbf{X}_{(\overline{S})} \,|\, \mathbf{Y}, S\Big] = \mathbf{0}$ for the complement of S. In order to evaluate the non-zero sub-matrix $\mathbb{E}[\mathbf{X}_{(S)} \,|\, \mathbf{Y}, S]$ one can use Bayes' rule as follows,

$$p(\mathbf{X}_{(S)}|\mathbf{Y}, S) = \frac{p\Big(\mathbf{X}_{(S)} \,|\, S\Big)\, p\Big(\mathbf{Y} \,|\, \mathbf{X}_{(S)}, S\Big)}{p(\mathbf{Y}|S)}, \tag{6.4}$$

where $p(\mathbf{Y}|S)$ is a normalizing constant for fixed $\mathbf{Y}$ and S. Moreover,

$$\mathbf{Y} \,|\, \mathbf{X}_{(S)}, S = \mathbf{A}_S\mathbf{X}_{(S)} + \mathbf{V},$$

where $\mathbf{A}_S$ is an $m \times k$ matrix restricted to those columns of $\mathbf{A}$ that are indexed by S. For a given $\mathbf{X}_{(S)}$, we have that $\mathbf{Y} \,|\, \mathbf{X}_{(S)}, S \sim \mathcal{MCN}_{m,q}\left(\mathbf{A}_S\mathbf{X}_{(S)}, \sigma_v^2\mathbf{I}_m, \mathbf{I}_q\right)$, so

$$p\Big(\mathbf{Y} \,|\, \mathbf{X}_{(S)}, S\Big) = \frac{1}{(\pi\sigma_v^2)^{mq}} \exp\Big(-\frac{1}{\sigma_v^2}\left\|\mathbf{Y} - \mathbf{A}_S\mathbf{X}_{(S)}\right\|_F^2\Big). \tag{6.5}$$

Ignoring the normalizing constant $p(\mathbf{Y}|S)$ and using the closed-form expressions of $p(\mathbf{X}_{(S)}|S)$ and $p(\mathbf{Y} \,|\, \mathbf{X}_{(S)}, S)$ from (6.2) and (6.5), we can rewrite (6.4) as

$$p(\mathbf{X}_{(S)}|\mathbf{Y}, S) \propto \exp\left(-\frac{\left\|\mathbf{X}_{(S)}\right\|_F^2}{\sigma_x^2} - \frac{\left\|\mathbf{Y} - \mathbf{A}_S\mathbf{X}_{(S)}\right\|_F^2}{\sigma_v^2}\right).$$

Since the prior $p(\mathbf{X}_{(S)}|S)$ and the likelihood $p(\mathbf{Y}|\mathbf{X}_{(S)}, S)$ are matrix variate complex normal with known variance, the posterior $p(\mathbf{X}_{(S)}|\mathbf{Y}, S)$ will also be matrix variate complex normal which is a symmetric unimodal distribution. Therefore, the mean of the posterior will be equal to its mode,

$$\mathbb{E}\Big[\mathbf{X}_{(S)}|\mathbf{Y}, S\Big] = \underset{\mathbf{X}_{(S)}\in\mathbb{C}^{k\times q}}{\operatorname{argmax}} \log p(\mathbf{X}_{(S)} \,|\, \mathbf{Y}, S).$$

The above convex optimization problem can be solved by setting the matrix gradient of the objective function to $\mathbf{0}$ and solving the resulting set of equations. This results in the following closed-form solution:

$$\mathbb{E}[\mathbf{X}_{(S)}|\mathbf{Y}, S] = \left(\mathbf{A}_S^{\mathrm{H}}\mathbf{A}_S + \frac{1}{\gamma}\mathbf{I}_k\right)^{-1}\mathbf{A}_S^{\mathrm{H}}\mathbf{Y}, \tag{6.6}$$

where $\mathbf{A}_S^{\mathrm{H}}$ denotes the conjugate transpose of $\mathbf{A}_S$.

For a fixed $\mathbf{Y}$ and all $S \in \Omega$, both $p(\mathbf{Y})$ and $p(S)$ are constant. Therefore, from Bayes' rule we can conclude that $p(S|\mathbf{Y}) \propto p(\mathbf{Y}|S)$. Moreover,

$$p(\mathbf{Y}|S) = \int_{\mathbf{X}_{(S)}\in\mathbb{C}^{k\times q}} p\Big(\mathbf{Y}, \mathbf{X}_{(S)}|S\Big)\mathrm{d}\mathbf{X}_{(S)} \propto$$
$$\int \exp\left\{-\frac{\left\|\mathrm{vec}(\mathbf{X}_{(S)})\right\|^2}{\sigma_x^2} - \frac{\left\|\mathrm{vec}(\mathbf{Y} - \mathbf{A}_S\mathbf{X}_{(S)})\right\|^2}{\sigma_v^2}\right\}\mathrm{d}\mathbf{X}_{(S)} \tag{6.7}$$

where the integration is over $\mathbf{X}_{(S)} \in \mathbb{C}^{k\times q}$. Since $\mathrm{vec}(\mathbf{Y} - \mathbf{A}_S\mathbf{X}_{(S)}) = \mathrm{vec}(\mathbf{Y}) - (\mathbf{I}_q \otimes \mathbf{A}_S)\mathrm{vec}(\mathbf{X}_{(S)})$, where $\otimes$ denotes the Kronecker product, the integral in (6.7) simplifies to

$$p(\mathbf{Y}|S) \propto w_S = \exp\left(\frac{\mathrm{vec}(\mathbf{A}_S^{\mathrm{H}}\mathbf{Y})^{\mathrm{H}}\mathbf{Q}_S^{-1}\mathrm{vec}(\mathbf{A}_S^{\mathrm{H}}\mathbf{Y})}{\sigma_v^4} + \log\Big(\det(\mathbf{Q}_S^{-1})\Big)\right),$$

where

$$\mathbf{Q}_S = \frac{1}{\sigma_v^2}\mathbf{I}_q \otimes \mathbf{A}_S^{\mathrm{H}}\mathbf{A}_S + \frac{1}{\sigma_x^2}\mathbf{I}_{kq}.$$

The above result is derived using similar arguments as those used in [29, p. 214, 215] for the real-valued SMV model ($q = 1$). Finally, the posterior $p(S|\mathbf{Y})$ of S, which is proportional to its likelihood $p(\mathbf{Y}|S)$, is obtained by normalizing w_S, i.e.,

$$p(S|\mathbf{Y}) = \frac{w_S}{\sum\limits_{\widetilde{S}\in\Omega} w_{\widetilde{S}}}. \tag{6.8}$$

6.2 Approximating the MMSE Estimator

Now we develop RandSOMP algorithm for approximating the MMSE estimator. The MMSE estimate of $\mathbf{X}$ given in (6.3) can be expressed in closed-form using (6.6) and (6.8). But the summation in (6.3) needs to be evaluated for all possible $S \in \Omega$,

Algorithm 11: RandSOMP algorithm (complex-valued case) [13]

Input: $\mathbf{A} \in \mathbb{C}^{m \times n}$, $\mathbf{Y} \in \mathbb{C}^{m \times q}$, k, L, γ, σ_v^2
for $l \leftarrow 1$ **to** L **do**
  $S^{(0)} \leftarrow \varnothing$, $\widetilde{\mathbf{X}}^{(0)} \leftarrow \mathbf{0}$
  for $i \leftarrow 1$ **to** k **do**
    $\mathbf{R} \leftarrow \mathbf{Y} - \mathbf{A}\widetilde{\mathbf{X}}^{(i-1)}$
    Draw an integer j randomly with probability proportional to

$$w_j = \exp\left\{ \frac{\gamma}{\sigma_v^2} \cdot \frac{\|\mathbf{R}^{\mathrm{H}} \mathbf{a}_j\|^2}{c_j} - q \log c_j \right\}$$

    $S^{(i)} \leftarrow S^{(i-1)} \cup \{j\}$
    $\widetilde{\mathbf{X}}^{(i)}_{(S^{(i)})} \leftarrow \left(\mathbf{A}^{\mathrm{H}}_{S^{(i)}} \mathbf{A}_{S^{(i)}} + \frac{1}{\gamma} \cdot \mathbf{I}_i \right)^{-1} \mathbf{A}^{\mathrm{H}}_{S^{(i)}} \mathbf{Y}$
    $\widetilde{\mathbf{X}}^{(i)}_{(\overline{S}^{(i)})} \leftarrow \mathbf{0}$
  end
  $\widehat{\mathbf{X}}^{(l)} \leftarrow \widetilde{\mathbf{X}}^{(k)}$
end
Output: $\widehat{\mathbf{X}}_{\mathrm{AMMSE}} \leftarrow \frac{1}{L} \sum_{l=1}^{L} \widehat{\mathbf{X}}^{(l)}$

which is computationally infeasible due to the large size of Ω ($|\Omega| = \binom{N}{K}$). One could approximate the MMSE estimate by randomly drawing a small number of sample row supports according to $p(S|\mathbf{Y})$ and evaluating the summation in (6.3) over these row supports. Again, due to the large size of the sample space Ω, this approach is not feasible. Next we describe how one can overcome these limitations via SOMP-like greedy pursuit. We propose to draw row supports iteratively by sampling one element at a time from a much smaller space. This results in row supports that are drawn from an approximation of $p(S|\mathbf{Y})$.

Suppose the rowsparsity of $\mathbf{X}$ is one, i.e., $|S| = k = 1$. This implies that $\Omega = \{1, 2, \ldots, n\}$, and

$$\mathbf{Q}_{S=\{j\}} = (c_j/\sigma_x^2) \cdot \mathbf{I}_q \quad \text{for} \quad j \in \Omega,$$

where $c_j = 1 + \gamma \cdot \|\mathbf{a}_j\|^2$ and $\mathbf{a}_j$ denotes the jth column of $\mathbf{A}$. Furthermore, $p(S = \{j\}|\mathbf{Y}) \propto w_{S=\{j\}} = w_j$ is given by

$$w_j = \exp\left\{ \frac{\gamma}{\sigma_v^2} \cdot \frac{\|\mathbf{Y}^{\mathrm{H}} \mathbf{a}_j\|^2}{c_j} - q \log c_j \right\}. \tag{6.9}$$

Since the size of the sample space $|\Omega| = n$, drawing a random row support S becomes trivial. For the case when $|S| = k > 1$, we propose an iterative greedy procedure that looks similar to SOMP and builds up the random row support iteratively one element at a time (see Algorithm 11). In the first iteration, we randomly draw an

element j_1 of S with probability proportional to the weights given in (6.9). We then compute the MMSE estimate $\widetilde{\mathbf{X}}$ of $\mathbf{X}$ assuming $S = \{j_1\}$ and the residual error matrix $\mathbf{R} = \mathbf{Y} - \mathbf{A}\widetilde{\mathbf{X}}$. In the second iteration we modify the weights in (6.9) by substituting the matrix $\mathbf{Y}$ by $\mathbf{R}$. We randomly draw another element j_2 of the row support using the modified weights and compute the MMSE estimate assuming $S = \{j_1, j_2\}$. This process continues for k iterations. After k-th iteration we have a randomly drawn row support $S = \{j_1, j_2, \ldots, j_k\}$ from a distribution that approximates $p(S|\mathbf{Y})$.

To reduce the estimation error, we run the above greedy procedure L number of times and average the results. Let $\Omega_\star$ denote the set of L row supports obtained from L independent runs. The *approximate MMSE estimate* of $\mathbf{X}$ is then

$$\widehat{\mathbf{X}}_{\text{AMMSE}} = \frac{1}{L} \sum_{S \in \Omega_\star} \mathbb{E}\Big[\mathbf{X} \,|\, \mathbf{Y}, S\Big]. \tag{6.10}$$

The posterior probability mass $p(S|\mathbf{Y})$ of each row support does not appear explicitly in the above formula. This is because the row supports having high probability are more likely to be selected in the sampling process than the row supports with low probability, so the row supports are represented in $\Omega_\star$ in (approximate) proportion to their probability masses. Hence, to approximate (6.3) simple averaging suffices.

In general, the approximate MMSE estimate of $\mathbf{X}$ given in (6.10) will not be rowsparse. In order to obtain rowsparse approximate MMSE estimate, we use an approach similar to the one used in RandOMP [23]. Let $H_k(\widehat{\mathbf{X}}_{\text{AMMSE}})$ denote the matrix obtained by setting all but the k largest (in terms of their ℓ_2-norm) rows of $\widehat{\mathbf{X}}_{\text{AMMSE}}$ to $\mathbf{0}$. Let $\widetilde{S}$ be the row support of $H_k(\widehat{\mathbf{X}}_{\text{AMMSE}})$, i.e., $\widetilde{S} = \text{rsupp}\Big(H_k(\widehat{\mathbf{X}}_{\text{AMMSE}})\Big)$. A k-rowsparse estimate of $\mathbf{X}$ is then obtained as

$$\widehat{\mathbf{X}}_{\text{rowsparse}} = \mathbb{E}\Big[\mathbf{X} \,|\, \mathbf{Y}, \widetilde{S}\Big]. \tag{6.11}$$

6.3 Empirical Results

Next we provide simulation results comparing the performance of SOMP and RandSOMP for complex-valued data. In order to illustrate the fact that the joint recovery of the rowsparse matrix in the MMV model is a much more effective approach, we include the results of the case where RandOMP is used to recover the individual sparse vectors one by one. It is demonstrated that RandSOMP performs better than RandOMP both in terms of normalized mean-squared error (MSE) and probability of exact support recovery. RandSOMP also outperforms SOMP in terms of normalized MSE, which is expected since RandSOMP approximates the MMSE estimate. More importantly, RandSOMP also outperforms SOMP in terms of probability of exact support recovery.

The simulation set-up is as follows. We generate $q = 30$ sparse vectors each of length $n = 300$ which share the same randomly chosen support of cardinality $k = 20$. All non-zero entries of the sparse vectors are independently drawn from the standard complex normal distribution (i.e., $\sigma_x^2 = 1$). The elements of the measurement matrix $\mathbf{A} \in \mathbb{C}^{150\times 300}$ (i.e., $m = 150$) are independently drawn from the standard

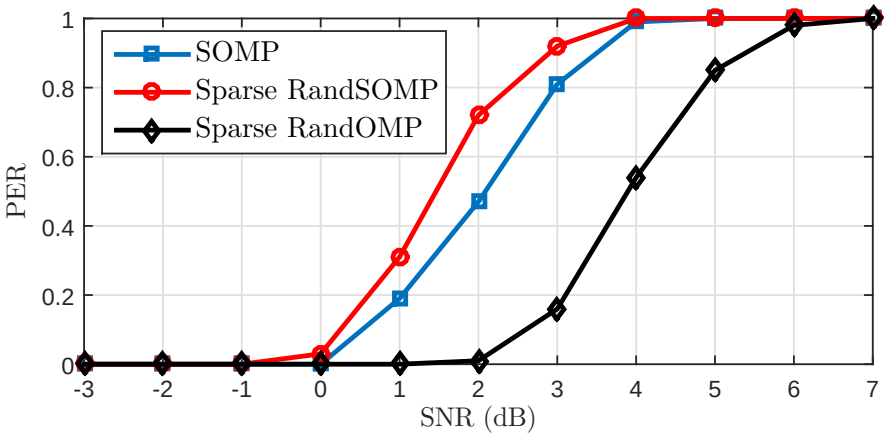


Figure 6.1: PER rates vs SNR. The parameter values are: $m = 150$, $n = 300$, $k = 20$, $L = 15$, $q = 30$. Sparse RandSOMP has higher PER rates for all the given SNR levels.

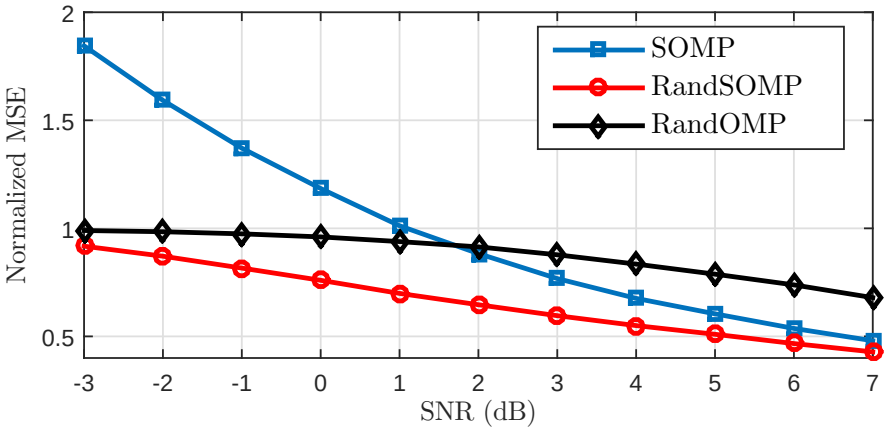


Figure 6.2: Normalized MSE vs SNR. The parameter values are: $m = 150$, $n = 300$, $k = 20$, $L = 15$, $q = 30$. RandSOMP has lower normalized MSE for all the given SNR levels.

complex normal distribution and each column is normalized to have unit norm. The measurements are contaminated by zero mean additive complex Gaussian noise with variance σ_v^2. The SNR is defined as $\gamma = \sigma_x^2/\sigma_v^2$. The number of randomly drawn row supports in both RandSOMP and RandOMP is $L = 15$. The experiments are averaged over 100 realizations.

First, the recovery of the correct signal support is considered. Figure 6.1 depicts the empirical *probability of exact (support) recovery (PER)* rates as a function of SNR. Since RandSOMP and RandOMP produce estimates that are not necessarily rowsparse, PER rates are computed for the sparsified estimates given in (6.11). As can be seen, the proposed RandSOMP algorithm outperforms both SOMP and RandOMP. Figure 6.2 depicts the empirical normalized MSE ($\|\widehat{\mathbf{X}} - \mathbf{X}\|_F^2 \big/ \|\mathbf{X}\|_F^2$). Again RandSOMP has the best performance at all SNR levels. One can also observe that in low SNR regime, the randomized algorithms are performing much better than

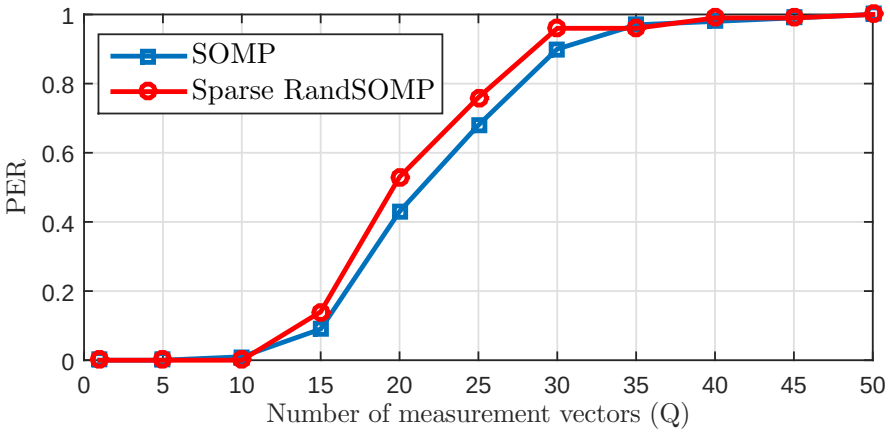


Figure 6.3: PER rates vs q. The parameter values are: $m = 150$, $n = 300$, $k = 20$, $L = 15$, SNR $= 3$ dB. Sparse RandSOMP has higher PER rates for all the given values of q.

the non-randomized SOMP. With increasing SNR, the performance of RandOMP improves much more slowly in comparison to RandSOMP. The performance of SOMP improves much more sharply, and at 2 dB SNR SOMP performs better than RandOMP. As expected, RandSOMP has the best performance since it is an approximate MMSE estimate.

Next we fix the SNR value at 3 dB and vary the number of measurement vectors q. Figure 6.3 depicts the PER as a function of q. When $q = 1$, the MMV model is reduced to the SMV model and thus SOMP and RandSOMP become equivalent to OMP [9] and RandOMP respectively. As q increases we expect to gain some improvement through joint processing of multiple measurement vectors. This is evident from Figure 6.3. When $q = 1$, the PER rate is near 0 for both SOMP and RandSOMP. With increasing q the PER rate increases and reaches full PER $(= 1)$ at $q = 50$. Again observe that for almost all the employed values of q, the proposed RandSOMP algorithm performs better than SOMP.

Chapter 7

Applications

Compressed sensing and sparse signal modeling has found applications in many diverse fields of engineering including medical imaging, radar and array signal processing, wireless communications and sensor networks, image processing, pattern recognition, and machine learning [29, 36, 37, 38]. In this chapter we will focus on two of these application areas. First we will apply the multiple measurement vectors (MMV) model formulation to direction-of-arrival (DOA) estimation using sensor arrays. Later we will apply the MMV model to RGB image denoising. We will use the simultaneous orthogonal matching pursuit (SOMP) algorithm and its Bayesian counterpart (RandSOMP) for solving the signal recovery problem in these applications. We will give a performance comparison of the two algorithms. The results illustrate the performance gain of RandSOMP over SOMP.

7.1 Direction-of-arrival Estimation using Sensor Arrays

In a typical DOA estimation setting, plane-wave signals from multiple sources are received and sampled at an array of omnidirectional sensors. From the measured samples it is then required to estimate the directions-of-arrival (DOAs) of the source signals. DOA estimation using sensor arrays is a widely studied problem and many different techniques have been proposed in the literature to solve this problem. In [39], the authors formulated DOA estimation as a joint sparse signal recovery problem and solved it using convex optimization. In this section, we follow the same problem formulation but instead use the SOMP and RandSOMP algorithms for finding the solution. First we will describe the problem formulation and results of a simulation study will be presented later.

We consider a uniform linear array (ULA) of m sensors that receive plane-wave signals from k sources with $k < m$. At time instant t, the output of the array can be modeled as $\mathbf{y}(t) = \mathbf{A}(\boldsymbol{\theta})\tilde{\mathbf{x}}(t) + \mathbf{v}(t)$, where $\mathbf{y}(t) \in \mathbb{C}^m$ is the array output, $\boldsymbol{\theta} \in \mathbb{R}^k$ is the unknown vector of true DOAs, $\mathbf{A}(\boldsymbol{\theta}) \in \mathbb{C}^{m \times k}$ is the measurement matrix whose columns consist of the array's steering vectors, $\tilde{\mathbf{x}}(t) \in \mathbb{C}^k$ is the unknown vector of source signals, and $\mathbf{v}(t) \in \mathbb{C}^m$ is the unknown additive noise. The steering vectors of

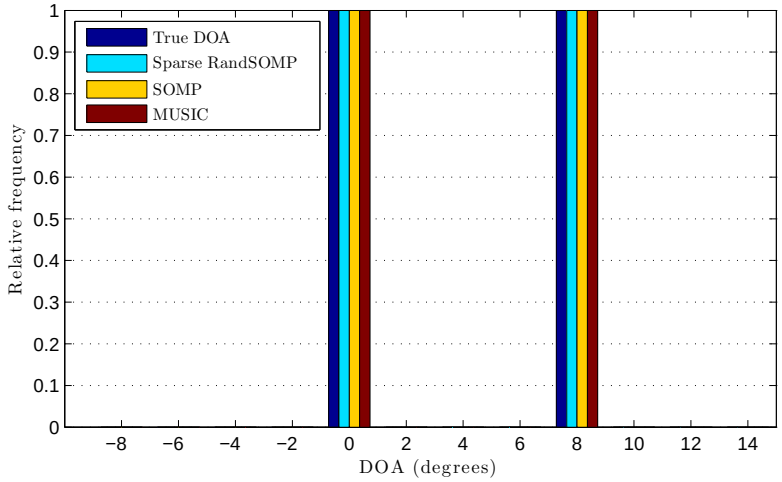


Figure 7.1: Relative frequency vs DOA (degrees). The parameter values are: $m = 20$, $n = 90$, $q = 50$, $k = 2$, $L = 10$, SNR $= -5$ dB, true DOAs at $0°$ and $8°$. Each algorithm has correctly estimated the true DOAs in every trial.

the array depend both on array geometry and the DOA vector $\boldsymbol{\theta}$. For a ULA, the i-th steering vector is given by $\mathbf{a}_i(\theta_i) = (1 \;\; e^{-j\pi \sin(\theta_i)} \ldots \; e^{-j\pi(m-1)\sin(\theta_i)})^{\mathrm{T}}$, where θ_i denotes the i-th component of $\boldsymbol{\theta}$. Since $\boldsymbol{\theta}$ is not known, $\mathbf{A}(\boldsymbol{\theta})$ will also be unknown and hence the compressed sensing formulation cannot be directly applied to the sensor array. In [39], the authors reformulate the sensor array model by using a fixed measurement matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ whose columns are formed from the steering vectors corresponding to a set of n DOA values ($n > m$) taken uniformly on a predefined sampling grid. Assuming the true DOA values lie on the grid, the sensor array output can be modeled as

$$\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{v}(t), \tag{7.1}$$

where $\mathbf{x}(t) \in \mathbb{C}^n$ is a k-sparse vector whose non-zero entries correspond to the true source signals in $\tilde{\mathbf{x}}(t)$ and the support of $\mathbf{x}(t)$ corresponds to the locations of the true DOA values on the sampling grid.

The model given in (7.1) is the familiar compressed sensing model. Furthermore, given multiple array snapshots $\mathbf{y}(t)$, $t = t_1, t_2, \ldots, t_q$, which are placed as columns of a matrix $\mathbf{Y} \in \mathbb{C}^{m \times q}$, one can write

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{V}, \tag{7.2}$$

where $\mathbf{X} \in \mathbb{C}^{n \times q}$ is a rowsparse matrix whose i-th column is equal to $\mathbf{x}(t_i)$, and $\mathbf{V} \in \mathbb{C}^{m \times q}$ is the noise matrix. Since the array output is now represented as the MMV model in (7.2), one can use SOMP and RandSOMP in finding the unknown source signals and their DOAs.

Now we will describe the simulation setup used for evaluating the performance of SOMP and RandSOMP in DOA estimation. In addition, we also test the performance of the MUSIC algorithm and use its results as a reference. We consider a ULA of $m = 20$ sensors that receive signals from $k = 2$ sources. The sources are (spatially and

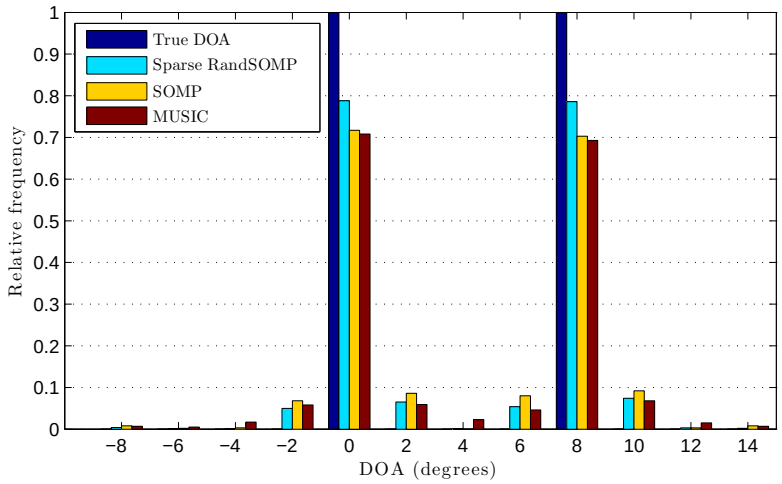


Figure 7.2: Relative frequency vs DOA (degrees). The parameter values are: $m = 20$, $n = 90$, $q = 50$, $k = 2$, $L = 10$, SNR $= -15$ dB, true DOAs at $0°$ and $8°$. Sparse RandSOMP has correctly found the true DOAs more frequently than SOMP and MUSIC.

temporally) independent standard complex normal random variables (i.e., $\sigma_x^2 = 1$). The true DOAs of the sources are $0°$ and $8°$. The sampling grid of $n = 90$ DOA values is defined in the interval $[-90°, 88°]$ with $2°$ step size, which means the true DOA values lie exactly on the grid. A total of $q = 50$ time samples are collected. The observations are corrupted by an additive white Gaussian noise with signal-to-noise ratio (SNR) equal to -5 dB. The number of randomly drawn row supports in RandSOMP is set at $L = 10$. The results of the simulation are based on 1000 trials. In each trial, each of the three algorithms provides an estimate of the DOAs. The total number of times a certain DOA value in the grid is estimated divided by the number of trials is called the relative frequency of the estimated value. In Figure 7.1, we show the relative frequencies of the estimated values given by SOMP, Sparse RandSOMP and MUSIC. As can be seen, all three algorithms have correctly estimated the true DOAs in all the trials resulting in a relative frequency equal to 1 at true DOAs ($0°$ and $8°$).

To test the performance of the three algorithms under more challenging conditions, we reduce the SNR to -15 dB. The values of the other parameters remain the same as before. The results are shown in Figure 7.2. It turns out that none of the three algorithms can estimate the true DOAs with 100% accuracy. Nevertheless, Sparse RandSOMP correctly estimates the true DOAs more frequently than its competitors. The relative frequencies at true DOAs obtained from RandSOMP are 7 to 8 percentage points higher than those provided by SOMP and MUSIC.

Next we test the performance of the algorithms when one of the true DOAs does not lie exactly on the sampling grid. We use the same parameter values as those used in generating the results in Figure 7.1, except the true DOAs are now $0°$ and $7.2°$. The resulting relative frequencies of the estimated values are shown in Figure

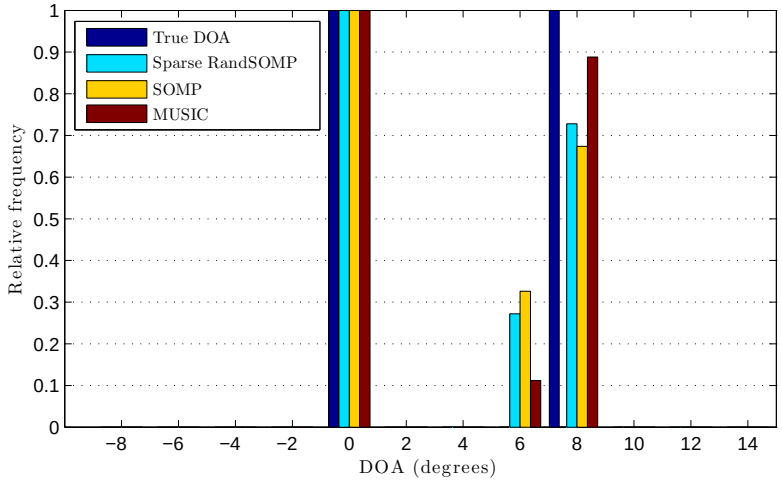


Figure 7.3: Relative frequency vs DOA (degrees). The parameter values are: $m = 20$, $n = 90$, $q = 50$, $k = 2$, $L = 10$, SNR $= -5$ dB, true DOAs at $0°$ and $7.2°$. The source at $7.2°$ is incorrectly located more frequently at $8°$ and less frequently at $6°$.

7.3. As expected, relative frequency of the estimated value at $0°$ is 1 for all three algorithms. In contrast, the DOA estimate of the source at $7.2°$ alternates between $6°$ and $8°$, with the latter value estimated more frequently than the former. This is due to the limitation imposed by the finite resolution of the sampling grid, i.e., SOMP and RandSOMP can only report DOA values that lie exactly on the grid.

One natural way of addressing the problem of finite grid resolution is to use a higher resolution grid but this results in greater computational load. A smarter way would be to estimate DOAs in multiple passes, i.e., first estimating the DOAs using a coarse-scale grid and then subsequently refining the grid only around those DOA values that are positively indicated in the previous passes. This strategy ensures that the grid resolution is increased only where needed, resulting in a more efficient computation.

7.2 Image Denoising

Real-world signals and images often possess some structure that distinguishes them from random noise. For example, images tend to lie on a lower-dimensional subspace embedded in a higher-dimensional space whereas random noise tends not to be restricted to any lower-dimensional subspace. Such distinguishing features make it possible to carry out many useful image processing tasks such as image denoising and compression. In this section we will focus on image denoising. We will show how sparse signal modeling can be successfully applied to image representation and demonstrate the effectiveness of SOMP and RandSOMP in removing additive Gaussian noise from RGB color images.

An RGB image of size $I \times J \times 3$ is a 3-dimensional array, where each $I \times J$ matrix slice represents one of the three color bands (Red, Green, Blue). We represent an

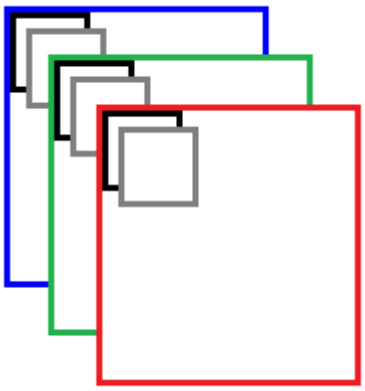


Figure 7.4: Two overlapping patches (black and gray) shown within a larger RGB image.

RGB image as a matrix $\mathbf{U} \in \mathbb{R}^{m\times 3}$, where $m = I \times J$ and each column of $\mathbf{U}$ is formed from vectorizing one of the three component matrices. The image $\mathbf{U}$ is corrupted by additive Gaussian noise $\mathbf{V}$ to give the noisy image $\mathbf{Y}$,

$$\mathbf{Y} = \mathbf{U} + \mathbf{V}, \tag{7.3}$$

such that both $\mathbf{U}$ and $\mathbf{V}$ are unknown and the goal is to recover the noise-free image $\mathbf{U}$ from its noisy version $\mathbf{Y}$.

As stated earlier, real-world images tend to lie on a lower-dimensional subspace. Consequently, there exist sets of orthonormal basis vectors (unitary dictionaries) such that only a small number of basis vectors are needed to reconstruct the images. Karhunen-Loève transform (KLT), discrete cosine transform (DCT), and many flavors of discrete wavelet transform (DWT) are examples of transformations with unitary dictionaries. By placing the vectors of a dictionary into the columns of a unitary matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$, one can uniquely represent the image $\mathbf{U}$ as

$$\mathbf{U} = \mathbf{A}\mathbf{X},$$

where $\mathbf{X} \in \mathbb{R}^{m\times 3}$ is a rowsparse matrix whose non-zero rows locate the basis vectors in $\mathbf{A}$ that are required for representing $\mathbf{U}$.

The idea of image representation with a unitary dictionary can be extended to a representation with multiple unitary dictionaries. This is useful when an image contains diverse features that cannot be captured compactly with a single dictionary. By stacking together multiple unitary dictionaries $\mathbf{A}_i \in \mathbb{R}^{m\times m}$, we get a larger overcomplete dictionary $\mathbf{A} = [\mathbf{A}_1\ \mathbf{A}_2 \cdots \mathbf{A}_p] \in \mathbb{R}^{m\times mp}$. The image $\mathbf{U}$ will again have a rowsparse representation (i.e. $\mathbf{U} = \mathbf{A}\mathbf{X}$, $\mathbf{X} \in \mathbb{R}^{mp\times 3}$), but the representation will no longer be unique due to the fact that the underlying system of equations is underdetermined. The noisy image given in (7.3) can now be modeled as

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{V} = [\mathbf{A}_1\ \mathbf{A}_2 \cdots \mathbf{A}_p]\mathbf{X} + \mathbf{V}, \tag{7.4}$$

(a) (b)

(c) (d)

Figure 7.5: RGB image denoising with SOMP and RandSOMP ($L = 10$, $C = 0.99$). (a) Original image; (b) Noisy image (PSNR = 12 dB); (c) Denoised image with SOMP (PSNR = 22.57 dB); (d) Denoised image with RandSOMP (PSNR = 22.47 dB).

which is the familiar MMV model where both $\mathbf{A}$ and $\mathbf{Y}$ are known. Therefore, one can use SOMP and RandSOMP to recover the rowsparse representation $\mathbf{X}$ and the noise-free image can then be obtained from $\mathbf{X}$ as $\mathbf{U} = \mathbf{AX}$.

Now we will describe the simulation setup used for evaluating the performance of SOMP and RandSOMP in image denoising. We take a noise-free RGB image (Lena) of size 256×256 and add Gaussian noise to it. Instead of processing the entire noisy image at once, we take smaller 8×8 patches of the image and denoise these patches separately. This ensures that the size of the overcomplete dictionary does not become too large. The patches are taken in an overlapping manner to reduce the blocking artifacts in the final denoised image. This is shown in Figure 7.4. The two smaller squares within the larger one are the overlapping patches, which are processed independent of each other. The overcomplete dictionary $\mathbf{A} \in \mathbb{R}^{64\times 192}$ is formed by stacking together DCT-II, symlet-4, and coiflet-4 dictionaries.

We model each noisy patch $\mathbf{Y}_i \in \mathbb{R}^{64\times 3}$ as $\mathbf{Y}_i = \mathbf{AX}_i + \mathbf{V}_i$, where $\mathbf{X}_i \in \mathbb{R}^{192\times 3}$ is the unknown rowsparse representation of the noise-free patch $\mathbf{U}_i \in \mathbb{R}^{64\times 3}$, and $\mathbf{V}_i \in \mathbb{R}^{64\times 3}$ is the additive noise. The number of randomly drawn row supports in

RandSOMP is set at 10. The stopping criterion for both SOMP and RandSOMP is based on the Frobenius norm of the residual error matrix $\mathbf{R}$. The algorithms are stopped when $\|\mathbf{R}\|_F < \sqrt{64 \times 3} \cdot C \cdot \sigma_v$, where C is a tuning parameter and σ_v is the standard deviation of the noise. The performance of the algorithms is measured using peak signal-to-noise ratio (PSNR), defined as

$$\mathrm{PSNR} = 20 \log_{10} \left(\frac{\max\big(\mathbf{U}\big)}{\mathrm{RMSE}\big(\mathbf{U}, \widehat{\mathbf{U}}\big)} \right),$$

where $\mathbf{U}$ is the original noise-free image, $\widehat{\mathbf{U}}$ is the denoised image recovered from the noisy image, and RMSE stands for the root mean-squared error.

$$\mathrm{RMSE}\big(\mathbf{U}, \widehat{\mathbf{U}}\big) = \frac{1}{\sqrt{3m}} \|\mathbf{U} - \widehat{\mathbf{U}}\|_F.$$

In Figure 7.5, we show the denoised images given by SOMP and RandSOMP. The PSNR of the noisy image was set at 12 dB. The value of the tuning parameter C was set at 0.99. The variance σ_x^2 of $\mathbf{X}$, which is required to run the RandSOMP algorithm, was estimated from the rowsparse representation given by SOMP. It is clear from the figure that both SOMP and RandSOMP have produced images that are visually much better than the noisy image. The gain in PSNR value given by both of the algorithms is over 10 dB. In comparison to RandSOMP, SOMP performs slightly better. This could be due to the fact that RandSOMP assumes $\mathbf{X}$ to be a matrix variate normal random variable, which is an assumption that is not guaranteed to hold in this case. In such a situation one can expect some degradation in performance. Nevertheless, this degradation does not seem to be high as both SOMP and RandSOMP provide a significant amount of improvement in the PSNR value of the image.

Chapter 8

Conclusion

The field of compressed sensing has fundamentally changed the way we understand sampling theory. Compressed sensing exploits sparsity of the signals and hence makes sampling possible at lower than Nyquist rate without incurring a significant loss of information. In this thesis, we reviewed some of the most important theoretical results in compressed sensing. We also reviewed several signal recovery algorithms and developed a novel Bayesian algorithm for multichannel sparse signal recovery. We also considered direction-of-arrival estimation with sensor arrays and image denoising as applications of compressed sensing.

In compressed sensing, the minimum number of measurements required for accurate signal recovery is largely determined by the degree of sparsity in the signal and not by the signal dimension. The structure of the measurement matrix is also critical for accurate signal recovery and it turns out that random measurement matrices are the right choice in this regard. Signal recovery can be carried out using either convex optimization or greedy algorithms. In general, methods based on convex optimization provide higher accuracy but greedy methods tend to be computationally more efficient and hence are better suited to large scale problems. Furthermore, theoretical results for some of the greedy algorithms have shown upper bounds on their error. These error bounds are largely determined by both the compressibility of the unknown signal and the strength of the measurement noise.

Bayesian algorithms that take into account the prior knowledge of the signal distribution provide more accurate signal recovery than non-Bayesian algorithms. At the same time, they suffer higher degradation in performance when prior assumptions about the signal distribution do not hold. For multichannel sparse signal models, methods that jointly estimate the sparse signals provide better performance than the methods that try to recover the sparse signals individually. In cases where accurate prior probability models of the signals are available, multichannel Bayesian recovery methods are expected to perform better than non-Bayesian methods. An example of this is the RandSOMP algorithm developed in this thesis for approximating the MMSE estimator of the multichannel sparse vectors, which outperforms the widely-used SOMP algorithm in terms of mean-squared error and probability of exact support recovery.

8.1 Future Work

Bayesian algorithms presented in this thesis can be extended to the cases where the sparse signal and the noise have non-Gaussian prior distributions. This is especially important when dealing with heavy-tailed distributions, since the estimators developed under Gaussian assumption perform poorly in the presence of outliers. Another possible extension of the work presented here is to devise new techniques for better approximating the MMSE estimator of the sparse signal in both single-channel and multi-channel settings.

Bibliography

[1] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 489–509, 2006.

[3] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *Information Theory, IEEE Transactions on*, vol. 52, no. 12, pp. 5406–5425, 2006.

[4] E. Candes and M. Wakin, "An Introduction To Compressive Sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, March 2008.

[5] R. Baraniuk, "Compressive sensing," *IEEE signal processing magazine*, vol. 24, no. 4, 2007.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[8] J. A. Tropp, "Just relax: Convex programming methods for identifying sparse signals in noise," *Information Theory, IEEE Transactions on*, vol. 52, no. 3, pp. 1030–1051, 2006.

[9] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, vol. 1, pp. 40–44, Nov. 1993.

[10] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approximation*, vol. 13, pp. 57–98, Mar. 1997.

[11] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, pp. 265–274, Nov. 2009.

[12] D. Needell and J. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, May 2009.

[13] A. Ejaz, E. Ollila, and V. Koivunen, "Randomized Simultaneous Orthogonal Matching Pursuit," 2015, Manuscript under review.

[14] A. M. Tillmann and M. E. Pfetsch, "The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing," *Information Theory, IEEE Transactions on*, vol. 60, pp. 1248–1259, Feb. 2014.

[15] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM journal on computing*, vol. 24, no. 2, pp. 227–234, 1995.

[16] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization," *Proceedings of the National Academy of Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.

[17] E. J. Candes and T. Tao, "Decoding by linear programming," *Information Theory, IEEE Transactions on*, vol. 51, no. 12, pp. 4203–4215, 2005.

[18] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, pp. 589–592, May 2008.

[19] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, pp. 253–263, Dec. 2008.

[20] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, pp. 629–654, Dec. 2008.

[21] T. Blumensath and M. E. Davies, "Normalized iterative hard thresholding: Guaranteed stability and performance," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, pp. 298–309, Apr. 2010.

[22] P. Schniter, L. Potter, and J. Ziniel, "Fast bayesian matching pursuit," in *Proc. IEEE Information Theory and Applications Workshop, 2008*, San Diego, CA, Jan 27-Feb 1 2008, pp. 326–333.

[23] M. Elad and I. Yavneh, "A plurality of sparse representations is better than the sparsest one alone," *Information Theory, IEEE Transactions on*, vol. 55, pp. 4701–4714, Oct. 2009.

[24] R. Crandall, B. Dong, and A. Bilgin, "Randomized Iterative Hard Thresholding: A Fast Approximate MMSE Estimator for Sparse Approximations," preprint. [Online]. Available: http://math.arizona.edu/~rcrandall/RandIHT_preprint.pdf

[25] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *Information Theory, IEEE Transactions on*, vol. 53, pp. 4655–4666, Dec. 2007.

[26] M. Davenport and M. Wakin, "Analysis of orthogonal matching pursuit using the restricted isometry property," *Information Theory, IEEE Transactions on*, vol. 56, pp. 4395–4401, Sep. 2010.

[27] T. Zhang, "Sparse recovery with orthogonal matching pursuit under RIP," *Information Theory, IEEE Transactions on*, vol. 57, pp. 6215–6221, Sep. 2011.

[28] A. Bjorck, *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[29] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[30] M. A. Davenport, D. Needell, and M. B. Wakin, "CoSaMP with redundant dictionaries," in *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*. IEEE, 2012, pp. 263–267.

[31] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *Signal Processing*, vol. 86, pp. 572–588, Mar. 2006.

[32] M. Duarte and Y. Eldar, "Structured Compressed Sensing: From Theory to Applications," *Signal Processing, IEEE Transactions on*, vol. 59, no. 9, pp. 4053–4085, Sept 2011.

[33] Y. Eldar and H. Rauhut, "Average case analysis of multichannel sparse recovery using convex relaxation," *Information Theory, IEEE Transactions on*, vol. 56, no. 1, pp. 505–519, Jan 2010.

[34] M. E. Davies and Y. C. Eldar, "Rank awareness in joint sparse recovery," *Information Theory, IEEE Transactions on*, vol. 58, no. 2, pp. 1135–1146, 2012.

[35] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*. Chapman and Hall/CRC, 1999.

[36] A. Y. Carmi, L. S. Mihaylova, and S. J. Godsill, *Compressed Sensing & Sparse Filtering*. Springer-Verlag Berlin Heidelberg, 2014.

[37] Y. C. Eldar and G. Kutyniok, *Compressed Sensing Theory and Applications*. Cambridge University Press, 2012.

[38] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[39] D. Malioutov, M. Cetin, and A. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 3010–3022, 2005.